REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	31 May 1995	Final Report 1 Apr 93 - 31 Mar 95

4. TITLE AND SUBTITLE
An Architecture For Incremental Construction of Distributed, Heterogeneous Systems
Part I

5. FUNDING NUMBERS
DAAH04-93-C-0013

6. AUTHOR(S)
Wolf Kohn & John James, Intermetrics, Inc.
Anil Nerode, Director, Mathematical Sciences Institute, Cornell Univ.
Jagdish Chandra, U.S. Army Research Office

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Intermetrics, Inc.
One Pacific Plaza
7711 Center Avenue, Suite 615
Huntington Beach, CA 92647

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
U.S. Army Research Office
P.O. Box 12211
Research Triangle Park, NC 27709-2211

10. SPONSORING / MONITORING AGENCY REPORT NUMBER
ARO 30754.1-MA-SDI

11. SUPPLEMENTARY NOTES
The views, opinions and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other documentation.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*
We discuss the application of the Kohn-Nerode-James optimality result of nonlinear systems optimization to the problem of implementing high-safety, high-assurance systems. We state the central problem of feedback control in terms of the most general categories of models (e.g. both logical models and evolution models which are the subject of the field of hybrid systems). We discuss optimal solutions of these nonlinear problems in terms of implementing control programs for a generic architecture suitable for enterprise-wide coordination and control. Finally we provide definitions of enterprise-wide, closed-loop control problems for several large-scale systems using this architecture and discuss synchronization with enterprise coordination processes. We summarize results for a reference architecture being demonstrated for the Department of Defense.

14. SUBJECT TERMS	15. NUMBER OF PAGES 48
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

An Architecture for Incremental Construction of Distributed, Heterogeneous Systems

Final Technical Report (CLIN 0002AE)
Contract DAAH04-93-C-0013

Intermetrics, Inc.

An Architecture for Incremental Construction of Distributed, Heterogeneous Systems

Wolf Kohn,*John James† ; Intermetrics, Inc.
Anil Nerode¶; Director, Mathematical Sciences Institute, Cornell University
Jagdish Chandra; U.S. Army Research Office

Abstract: We discuss the application of the Kohn-Nerode-James optimality result for nonlinear systems optimization to the problem of implementing high-safety, high-assurance systems. We state the central problem of feedback control in terms of the most general categories of models (e.g. *both* logical models and evolution models which are the subject of the field of hybrid systems). We discuss optimal solutions of these nonlinear problems in terms of implementing control programs for a generic architecture suitable for enterprise-wide coordination and control. Finally we provide definitions of enterprise-wide, closed-loop control problems for several large-scale systems using this architecture and discuss synchronization with enterprise coordination processes. We summarize results for a reference architecture being demonstrated for the Department of Defense.

1. Introduction

The need for a hybrid systems approach: In this paper we elaborate on a generic software architecture for large-scale systems provided at the first Cornell conference on hybrid systems [9, 10] and discuss application of the architecture using a recent optimization result [37] to achieve a theoretical basis for *rigorous integration* of heterogeneous models. By heterogeneous models we mean the most general categories of models (e.g. *both* logical models and evolution models which are the subject of the field of hybrid systems). The term integration is commonly used to mean the *ad hoc* coupling of diverse systems, achieved largely through heuristics, engineering experience and experimentation. By *rigorous integration* we mean that, to the extent that the logical models are correct, the evolution models are correct, and the composition rules are correct, our approach for integration of heterogeneous models will *extract* a correct, integrated program. Moreover, this integration mechanism is computationally tractable and epsilon-optimal within the constraints of the qualitative and quantitative uncertainties of the composed problem.

We also assert that recent results in hybrid systems theory support this rigorous approach for integration of diverse models and provides a path to achieving high-safety, high-assurance computer-controlled systems. The basis for this assertion is use of a new approach for coupling of numeric and symbolic computation which supports ***unification*** of trusted models in a *single* mathematical model (the hybrid system state) which is analyzed in a single mathematical entity (the carrier manifold) instead of ***experimentation*** with changing *different kinds* of trusted models in at least *two kinds* of mathematical models (logic or evolution models) analyzed in fundamentally different mathematical entities (T0 topologies for logic models and Hausdorf for evolution models). These ***novel extensions*** (the *hybrid system state* and the *carrier manifold*) of the current control concepts of system state for evaluation of analysis results and Hausdorf topologies (or Hardy or Hilbert spaces) for construction of mathematical proofs of properties of system state ***support rigorous analysis*** of the most general kinds of control systems. Our approach for achieving this rigor is to rely on the mathematical foundations of a formal extraction process for construction of automata which simultaneously comply with current logical constraints and current

* partially sponsored by SDIO/IST grant DAAH04-93-C-0113
† partially sponsored by SDIO/IST grant DAAH04-93-C-0113
¶ supported in part by Army Research Office contract DAAL03-91-C-0027 and by DARPA-US Army AMMCCOM (Picatinny Arsenal, N. J) contract DAAA21-92-C-0013 to ORA Corp. and by SDIO/IST grant DAAH04-93-C-0113

continuum constraints [7, 8, 9, 10, 12, 13, 21, 37, 47]. In this paper we will interpret the impact a recent result concerning epsilon-optimal solution of the nonlinear Hamilton-Jacobi-Bellman equation has on a broad range of application areas from the point of view of a generic architecture for integration of heterogeneous components of enterprise-wide control systems [7, 8, 21, 37]. While the optimality result applies to systems in the most general sense, we limit our discussion here to high-safety, high-assurance systems.

What is a high-safety, high assurance system? We take the description to mean a distributed, goal-oriented system of computers, communication networks, and software applications, supervised by people interacting with the system to achieve safe, reliable performance in the presence of system uncertainties and disturbances. Examples include: transportation systems (like the Advanced Automation System under development for air travel or the Intelligent Vehicle Highway System being investigated for future highway systems); discrete manufacturing systems under development to achieve increased agility and higher quality, batch process manufacturing systems being developed to increase yield and quality of products; medical information systems being developed for support of telemedicine as well as patient administration, diagnosis and treatment; communication networks being developed for control of multimedia processes across international borders; large-scale power systems; and military command and control systems being developed to assist in control of the use of deadly force to increase combat effectiveness and reduce fratricide. These systems are such that *unsafe operation can cause loss of life* or *unreliable operation can cause loss of revenue.* Our goal is to guarantee approximately optimal control of the behavior of the closed loop system in the presence of both qualitative and quantitative uncertainties where "approximately optimal" control is weak or measure-valued control (in the sense of L. C. Young [15]).

The failure of the current technology: Control science and the control industry is currently incapable of delivering these large-scale, high-safety, high-assurance computer-controlled systems at reasonable cost. Indeed, there have been numerous recent failures in attempts to build large-scale, computer-controlled systems, even though the controls industry has decades of experience in successful implementation of small-scale systems. Dr. John Cassidy recently gave a plenary speech at the American Control Conference. In his presentation he observed that "... 77% of the software of control systems is for implementation of logic and scheduling and 23% of the software is for implementation of control algorithms. We still do not have a methodology for integrating logic and control algorithms."* Dr. Cassidy has extensive experience in managing control theory developments and supervising control implementations for General Motors, General Electric, and United Technologies. His experience is the basis for his observation of the existing split of control implementation software into two categories with the majority of the code being written for support of planning and scheduling functions and a significant portion being written to implement control algorithms. He elaborated at some length on the importance of resolving the current *lack of a methodology* for integration of logic and control algorithm software, the former based on set-based techniques, and the latter based on differential equation theory, control systems theory and analysis techniques. This split in implementation software is the direct result of a split in the mathematical models used for analysis and design of control systems.

Split in models: Logical models of system behavior (such as safety constraints or start-up and shut-down conditions or interface constraints into other system components) and cognitive models of human behavior are built using linguistic tools which depend on the set-based mathematics of algebraic topology. Analytical models of system behavior (such as motion and position) are built using analysis tools which depend on the continuum-based mathematics of differential operators. Experiments are necessary to determine the behavior of the composition of models for safety,

---

** Dr. John Cassidy, Director of Research, United Technologies Corporation, Plenary Speech, **Control Technology and the 21st Century**, American Control Conference, San Francisco, CA, 2 June 1993.

reliability and performance constraints over a broad range of logical conditions and continuum values. The general approach currently used for verification and validation of complex systems is to explore design failure modes and track correction of bugs in the software until a satisfactory level of performance (absence of failure for expected operating conditions) is reached and success is declared. Unfortunately, while it has been possible to experimentally evaluate more and more complex finite-state machines, it is not possible (and will never be possible) to experiment with all possible combinations of values of logical and continuum parameters for computer-controlled systems. This leaves a nagging expectation that not all of the states of the computer finite-state machine have been visited and tested with continuum values which might alter system performance. Also, when new capabilities are added, new failure modes are created so the whole testing process must begin again.

An architecture for incremental integration: This situation has led to interest by Landauer and Bellman in pursuing new mathematical foundations for large-scale systems [38]. We agree with Cassidy, Landauer and Bellman and submit that Kohn-Nerode hybrid systems theory is a rigorous methodology for achieving the required incremental integration of heterogeneous models. We believe that our recent (mathematically sound and complete) optimal control result, when combined with the multiple-agent hybrid control architecture and component generation tools currently under development, will provide a cost-effective engineering path for construction of large-scale, high-safety, high-assurance systems. In achieving this integration we also believe *synchronizing events and continuum values* (maintaining stable operation) across the broad range of time scales of computer-controlled systems and *synchronizing semantic agreement* between plans and implementations (maintaining consistent views) across the broad range of disciplines needed to build computer-controlled systems requires explicit support by a generic architecture. This is not a new position but a restatement of a long series of efforts [1, 2, 3, 4, 5, 6] focused primarily on providing a generic architecture for manufacturing enterprises but also used for other complex computer-controlled systems (such as mobile robots and military command and control).

In this paper we provide a summary of several applications of our Multiple-Agent Hybrid Control Architecture (MAHCA), [7, 8, 9, 10, 12, 13, 21, 37, 47] for achieving integration of logic and continuum models of system behavior. These examples are explained in the context of a recent result [see 37] (the Kohn-Nerode-James optimality result) for epsilon-optimal solution of the nonlinear Hamilton-Jacobi-Bellman (HJB) equation for a class of hybrid systems.

This paper discusses how our multiple-agent hybrid control architecture (MAHCA) is an appropriate framework for epsilon-optimal, closed-loop control of enterprise-wide activities. The paper is organized into five remaining sections. Section 2 provides a statement of the central problem of feedback control in terms of the MAHCA framework. Section 3 provides a discussion of MAHCA relative to other enterprise-wide architectures. Section 4 provides an overview of use of the architecture for control of qualitative and quantitative attributes of several large-scale enterprises. Section 5 provides a more detailed discussion of applying the architecture to a Department of Defense functional activity. Section 6 provides a summary.

2. Statement of the central problem of feedback control in terms of the MAHCA framework.

Epsilon optimality for hybrid systems is the result of: (1) explicit statements of qualitative and quantitative models of system components, system inputs, and disturbance uncertainties; (2) the desired qualitative and quantitative closeness constraints of the closed-loop system; and (3) generation of programs (automata) to force closed-loop compliance with those constraints. The problem we discuss is on-line modification of the evolution of the system:

$$\dot{\mathbf{X}}(t) = \mathbf{f}(\mathbf{X}(t), \mathbf{u}(t), t), \quad \text{with} \tag{1}$$

$$\left.\begin{array}{l}\mathbf{X}: I \to M \\ \mathbf{u}: I \to U\end{array}\right\} M, U \text{ manifolds, } \mathbf{u} \text{ measureable, } t \in I, \text{ interval of real line}$$

Our goal is to design a control law

$$\mathbf{u}(t) = \gamma(\mathbf{X}(t), t), \text{ which alters the closed-loop evolution} \tag{2}$$

$$\dot{\mathbf{X}}(t) = \mathbf{F}(\mathbf{X}(t), t) = f\big(\mathbf{X}(t), \gamma(\mathbf{X}(t), t), t\big), \text{ with} \tag{3}$$

$\dot{\mathbf{X}}(t) : I \to \mathrm{TM}_{\mathrm{X}}$, Tangent space at $\mathbf{X}$

$\mathrm{F} : M \times I \to$ TM, Tangent bundle

such that closed-loop qualitative and quantitative performance requirements are met.

2.1 Explicit statement of qualitative and quantitative models and disturbance uncertainties

We model the qualitative and quantitative behavior of the system using explicit rules which constrict the evolution of the sytem. The knowledge base of the controller is a composition of external rules of the form:

$$\mathbf{p}_i(G, S, A, E) \text{ if } \mathbf{e}_1^i(x_1, \ldots, x_n) \wedge \ldots \wedge \mathbf{e}_m^i(x_1, \ldots, x_n) \wedge \text{unify}\,(x_1, \ldots, x_n; G, S, A, E) \text{ where} \tag{4}$$

G: Goal variables, S: Sensor variables, A: Actuator variables, E: Evaluation variables

and internal rules of the form:

$$\begin{array}{l}\mathbf{q}_i(y_1, \ldots, y_m) \text{ if } \mathbf{e}_1^i(y_1, \ldots, y_m) \wedge \ldots \wedge \mathbf{e}_m^i(y_1, \ldots, y_m), \text{ where} \\ \mathbf{e}_i^j(y_1, \ldots, y_m) \text{ is of the form} \\ \mathbf{e}_i^j: \mathbf{w}_i^j(y_1, \ldots, y_m) = \mathbf{v}_i^j(y_1, \ldots, y_m) \text{ or} \\ \quad \mathbf{w}_i^j(y_1, \ldots, y_m) \neq \mathbf{v}_i^j(y_1, \ldots, y_m) \text{ or} \\ \quad \mathbf{w}_i^j(y_1, \ldots, y_m) \leq \mathbf{v}_i^j(y_1, \ldots, y_m) \text{ or} \\ \quad \left.\begin{array}{ll}\mathbf{p_k}(G, S, A, E) & \text{or} \\ \mathbf{q_k}(y_1, \ldots, y_m) & \end{array}\right\} \text{recursion}\end{array} \tag{5}$$

with the syntax of the clauses similar to the Prolog language,

$$\text{Head if Body} \tag{6}$$

The Head is a functional form, $\mathbf{p}(x_1, \ldots x_n)$, taking values in the binary set [*true, false*] with x_1, $x_2, \ldots x_n$ as variables in the domain D of the controller. The variables in the clause head are assumed to be universally quantified.

The Body of a clause is a conjunction of one or more logical terms,

$$e1 \wedge e2 \wedge \ldots \wedge em \tag{7}$$

The rules for are written to capture both qualitative and quantitative constraints on system operation. Enterprise process dynamics are modeled in terms of the conservation of the flow of one or more enterprise process variables through a distributed *logic communication network* of control agents. For example, in transportation networks, the flow of the sum of vehicles is conserved; in communication networks the flow of unsatisfied demand for network services is conserved; and in manufacturing the flow of product through the factory floor creates a demand for services at work cells (demand for short) which is synchronized through conservation of the flow of demand through the logic communication network. For communications enterprises, the service demand represents the demand for connections, bandwidth, or higher quality of service. For manufacturing enterprises the flow of unsatisfied demand between producers and consumers can be used to generate actions to implement factory planning, scheduling and control functions. For transportation systems the flow of vehicles and the (opposite direction) flow of highway voids can be used to generate programs for traffic control [48]. The architecture of the logical communications network is composed of connection links and agents. The agents are allocated at the nodes of the network. As a function of time (t), the network configuration varies because new agents are *spawned* and become part of the network or old ones drop out.

In equations 8 through 21 below we apply the Kohn-Nerode definition for continuity between points in a manifold to state the problem for epsilon-optimal control of multimedia processes associated with manufacturing [39]. Solution of the manufacturing problem is given by equations 22 and 23 [37, 39]. A discussion of application of the architecture to command and control of military operations is given in [21].

Let A_i, $i = 1, \ldots, N(t)$ denote the agents active at the current time t. At each time t, the status of each agent is given by a point in a locally differentiable manifold M [9]. The demand of an active agent A_i is given by a continuous* function D_i,

$$D_i : M \times T \rightarrow R^+ \tag{8}$$

Where T is the real line (time space) and R^+ is the positive real line. A point **p** in the manifold M is represented by a data structure of the form:

$$\mathbf{p}(\text{id}, \text{proc}(\text{proc\_data}), \text{media}(\text{media\_data}), \text{in}(\text{synch\_data}), \text{mp}(\text{mult\_data})) \tag{9}$$

where:

id is an identifier taking values in a finite set ID,

proc() is a relation characterizing manufacturing processes status; it depends on a list of parameters labeled *proc_data,* which define the load and timing characteristics of the process involved.

media is a relation that captures attributes of the multimedia process being represented; it depends on a list of parameters labeled *media_data*; these parameters characterize constraint instances, at that point, of the process being represented at a level of abstraction compatible with the logic communication network.

in() is a relation carrying synchronization information of the logic communication network with respect to the hierarchical organization of the network. Specifically, it characterizes the protocol at the operation point. This includes information such as priority level, connectivity and time constants.

mp() carries multiplicity information that is, it represents the level of network usability at this point. The associated parameter list, *mult_data,* is composed of statistical parameters reflecting the network's load.

---

* We will explain this terminology in the context of our model later on in this section.

The parameter lists in the data structure of the points of M, are composed of integers, such as number of users, reals, such as traffic loads, and discrete values such as process identifiers or switches. These values characterize the status of the network and the active processes. Computing the evolution of these parameters over time is the central task of the model.

The dynamics of the logic communication network is characterized by certain trajectories on the manifold M. These trajectories characterize the flow of information through the network and its status. Specifically, we need to define two items: 1) A *generator* for the demand functions :

$$\{D_i(p,t) \mid i \in I(t), p \in M\} \tag{10}$$

with I(t) the set of active agents at time t, and 2) the general structure of the functions in (3) for an *active* agent at time t:

$$D_i(p,t) = F_i(C_i^u, D, \alpha_i)(p,t) \tag{11}$$

Where F_i is a smooth function, D is the vector of demand functions, C_i^u is the unsatisfied demand function, and α_i is the command action issued by the ith agent. We will show below that these actions are implemented as infinitesimal transformations defined in M.

In general, a manifold M is a topological space (with topology Θ) composed of three items:

1) A set of points of the form of (9), homeomorphic to R^k with k an integer.
2) A countable family of open subsets of M, $\{U_j \mid j \text{ countable}\}$ such that:

   $\bigcup_j U_j = M$. and
3) A family of *smooth*** functions, $\{\varphi_j \mid \varphi_j : U_j \to V_j\}$, where V_j for each j is an open set in R^k . The sets U_j are referred to in the literature as coordinate neighborhoods or charts. For each chart the corresponding function φ_j is referred to as its coordinate chart. The coordinate chart must satisfy the condition that:

   Given any charts U_j, U_i such that $U_j \cap U_i \neq \varnothing$,

   the function $\varphi_i \circ \varphi_j^{-1} : \varphi_j(U_j \cap U_i) \to \varphi_i(U_j \cap U_i)$ is smooth.

* In the literature, one usually finds the Hausdorf property in the definition of manifolds [24]. Since this does not hold in our application, we will not discuss it.

To customize the generic definition of manifold to our application, we start with the topology Θ associated with M. The points of M have a definite structure characterized by the intervals of parameter values in *proc_data*, *media_data*, *synch_data* and *mult_data*. The number of these parameters equals k. The *Knowledge* about these parameters is incorporated into the model by defining a topology Ω on R^k [7].

** By smooth, we mean that they posses arbitrarily many continuous derivatives

The open sets in Ω are constructed from the clauses[1] encoding what we *know* about the parameters. The topology Θ of M is defined in terms of Ω as follows:

> for each open set W in Ω, such that $W \subseteq V_j \subseteq R^k$, we require that the set $\varphi_j^{-1}(W)$ be in Θ. These sets form a basis for Θ, so that $U \subset M$ if and only if for each $p \in U$ there is a neighborhood of this form contained in U; that is, $p \in \varphi_j^{-1}(W) \subset U$, with $\varphi_j : U_j \rightarrow V_j$ a chart containing p.

We amplify the results of the above discussion of the manifold toplogy in the next three figures which provide views of:

1. the one-to-one mapping between horn clauses written in the Equational Logic Language (see Appendix A for a discussion of ELL) and open sets in the manifold (Figure 1) such that each open set has a corresponding chart function,
2. the evolution from one open set to another open set in the manifold, with no conflict on the boundary (i.e. $\varphi_i\varphi_j^{-1}: U_j \cap U_i \rightarrow V_i \cap V_j$) (Figure 2), and
3. each chart can be grounded such that there is a correspondence to a suitable Euclidean space. Suppose there are *n* distinct parameters in the set of clauses and $\varphi_i: V_i \rightarrow V_i \subseteq R^{n_i}$, then the euclidean space is R^n (Figure 3).

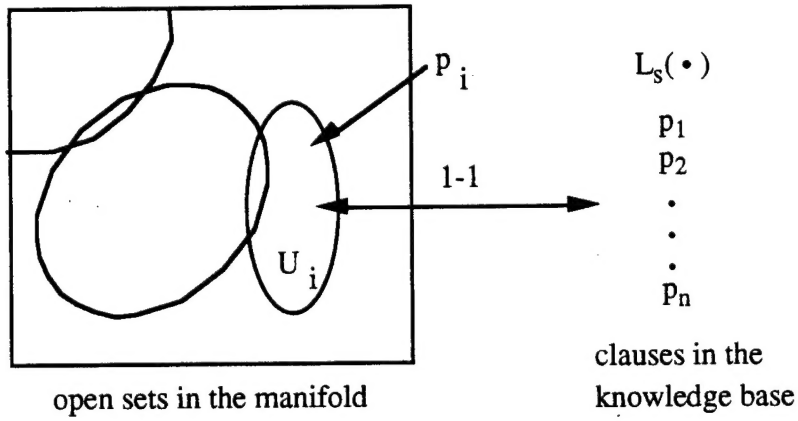
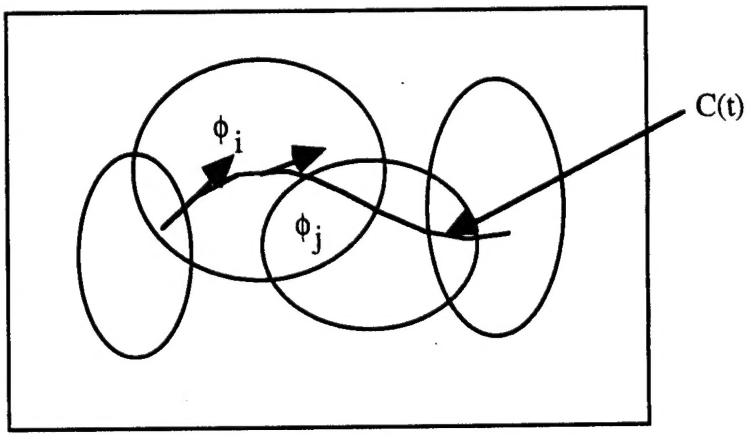


Figure 1. Every Rule has a 1-1 Correspondence with an Open Set



[1]We discuss these clauses in section 3.

Figure 2. No Conflict on Open Set Boundaries

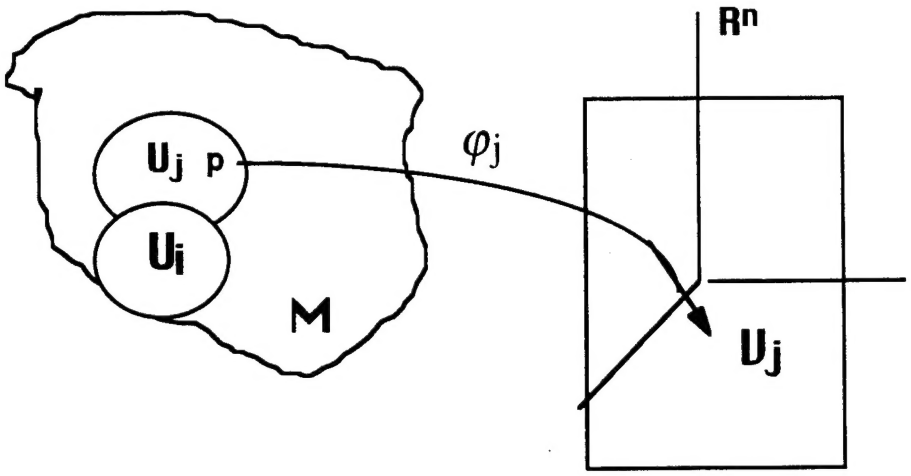


Figure 3. Each chart can be grounded such that there is a correspondence to a suitable Euclidean space

To characterize the actions commanded by the intelligent manufacturing controller, we need to introduce the concept of derivations on M. Let F_p be the space of real-valued smooth functions f defined near a point p in M. Let f, g be functions in F_p. A derivation of v of F_p is a map: $v : F_p \rightarrow F_p$, that follows linearity and Leibniz' rule:

$$v(f+g)(p) = \big(v(f) + v(g)\big)(p) \qquad \text{(Linearity)}$$

$$v(f \cdot g)(p) = \big(v(f) \cdot g + f \cdot v(g)\big)(p) \qquad \text{(Leibniz' Rule)}$$

Derivations define vector fields on M and a class of associated curves called integral curves [12]. Suppose that C is a smooth curve on M, parameterized by , $\phi : I \rightarrow M$ with I a subinterval of R. In local coordinates, $p= (p^1, \ldots, p^k)$, C is given by k smooth functions: $\phi(t) = \big(\phi^1(t), \ldots, \phi^k(t)\big)$ with derivative with respect to t given by : $\dot{\phi}(t) = \big(\dot{\phi}^1(t), \ldots, \dot{\phi}^k(t)\big)$. We introduce an equivalence relation on curves in M as the basis for the definition of tangent vectors at a point p in M [9]. Two curves $\phi_1(t)$ and $\phi_2(t)$ passing through p are said to be equivalent at p (notation: $\phi_1 \sim \phi_2$) , if they satisfy the following conditions:

For some t, τ in $I \subset R$

$\phi_1(t) = \phi_2(\tau) = p$, and

$\dot{\phi}_1(t) = \dot{\phi}_2(\tau)$

If $[\phi]$ an equivalence class containing ϕ , a tangent vector to $[\phi]$ is a derivation $v|_p$, defined in the local coordinates $(p^1, \ldots, p^k)$, by :

Let $f : M \rightarrow R$ be a smooth function. Then,

$$v|_p(f)(p) = \sum_{j=1}^{k} \dot{\phi}^j(t) \frac{\partial f(p)}{\partial p^j} \qquad \text{with} \qquad p = \phi(t) \tag{12}*$$

The set of tangent vectors associated with all equivalence classes at p defines a tangent vector space at p, denoted by TM_p. The set of tangent spaces associated with M can be "glued" together to form a manifold called the tangent bundle and denoted by TM:

$$TM = \bigcup_{p \in M} TM_p$$

We will explain explicitly how this glue is implemented, after we introduce the concept of a vector field and discuss its relevance to the model.

A vector field on M is an assignment of a derivation v to each point of $M : v|_p \in TM_p$, with $v|_p$ varying smoothly from point to point. In our model, we will always express vector fields in local coordinates. Let $(p^1, \ldots, p^k)$ be local coordinates then,

$$v|_p = \sum_j \lambda^j(p) \frac{\partial}{\partial p^j} \tag{13}$$

Comparing (12) and (13) we see that if $p = \phi(t)$ is a parameterized curve in M whose tangent vector at any point coincides with the value of v at the same point then,

$$\dot{\phi}(t) = v|_{\phi(t)}$$

for all t. In local coordinates, $p = (\phi^1(t), \ldots, \phi^k(t))$ must be a solution to the autonomous system of ordinary differential equations:

$$\frac{dp^j}{dt} = \lambda^j(p) \quad \text{for } j = 1, \ldots, k \tag{14}$$

In our manufacturing intelligent controller, each command is implemented as a vector field in M. Each agent in the controller constructs its command field as a combination of 'primitive' predefined vector fields. Since the chosen topology for M is not metrizable, given an initial condition, we cannot guaranty a unique solution to (14) in the classical sense. However, they have as solutions a class of continuous trajectories in M called Relaxed Curves [15], and on this class, the solutions to (14) are unique.

Here, we describe some of the properties of relaxed curves, as they relate to our manufacturing process model and control. For this objective, we need to introduce the concept of flows in M. If v is a vector field, the parameterized integral curve passing through a point p in M, denoted by $\Psi(t,p)$, is termed the flow generated by v. The flow satisfies the following properties:

---

* Although we have expressed the tangent vector in terms of the chosen local coordinates, it can be shown that this definition is independent of the chosen local coordinates [10].

$$\Psi(t, \Psi(\tau, p)) = \Psi(t+\tau, p) \quad \text{(semigroup property)}$$

$$\Psi(0, p) = p \quad \text{(initial condition)}$$

and

$$\frac{d}{dt}\Psi(t,p) = v\big|_{\Psi(t,p)} \quad \text{(flow generation)} \tag{15}$$

Now we are ready to customize these concepts for our model. Suppose that agent i in the communication network is active. Let $D > 0$ be the width of the current decision interval, $[t, t + \Delta)$. Let $U_i(p, t)$, $p \in M$ be the unsatisfied demand at the beginning of the interval. Agent i has a set of primitive actions:

$$\left[v_{i,j} \,\middle|\, j = 1, \cdots, n_i,\ v_{i,j}\big|_p \in TM_p, \text{ for each } p \in M \right] \tag{16}$$

During the interval, $[t, t+\Delta)$, agent i schedules one or more of these actions and determines the fraction, $\alpha_{i,j}(p,t)$, of D that action $v_{i,j}$ must be executed, as a function of the current service request, $S_{ri}(t,p)$, and the demand of the active agents in the logic communication network, $D(p,t) = \left[D_1(p,t), \cdots, D_{N(t)}(p,t)\right]$. Figure 4 illustrates a schedule of actions involving three primitives. We will use this example as means for describing the derivation of our model.

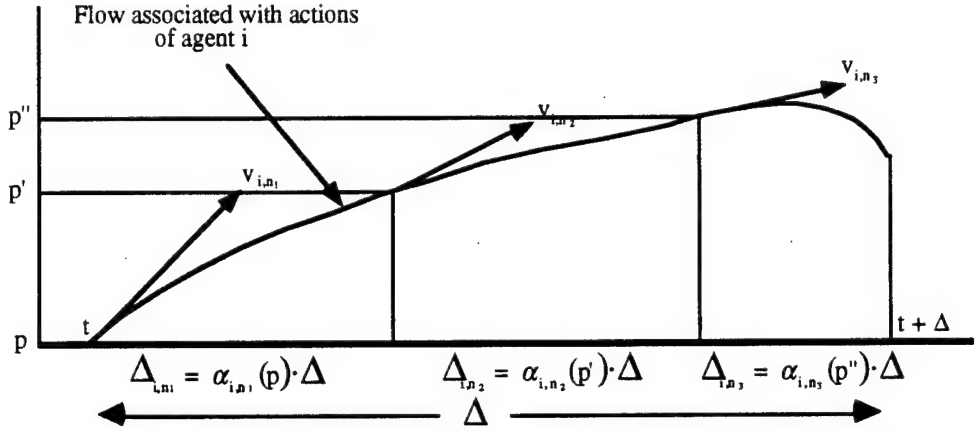


Figure 4. Conceptual illustration of agent action schedule

The flow Ψ_i associated with the schedule of Figure 4 can be computed from the flows associated with each of the actions:

$$\Psi_i(\tau, p) = \begin{cases} \Psi_{v_{i,n_1}}(\tau, p) \text{ if } t \le \tau < t + \Delta_{i,n_1} \\ \Psi_{v_{i,n_2}}\left(\tau, \Psi_{v_{i,n_1}}\left(t + \Delta_{i,n_1}, p\right)\right) \text{ if } t + \Delta_{i,n_1} \cdot \le \tau < t + \Delta_{i,n_1} + \Delta_{i,n_2} \\ \Psi_{v_{i,n_3}}\left(\tau, \Psi_{v_{i,n_2}}\left(t + \Delta_{i,n_1} + \Delta_{i,n_2}, \Psi_{v_{i,n_1}}\left(t + \Delta_{i,n_1}, p\right)\right)\right) \text{ if} \\ t + \Delta_{i,n_1} + \Delta_{i,n_2} \le \tau < t + \Delta_{i,n_1} + \Delta_{i,n_2} + \Delta_{i,n_3} \end{cases} \tag{17}$$

with $\Delta_{i,n_1} + \Delta_{i,n_2} + \Delta_{i,n_3} = \Delta$ and $\alpha_{i,n_1} + \alpha_{i,n_2} + \alpha_{i,n_3} = 1$

We note that the flow Ψ_i, given by (10), characterizes the evolution of the process as viewed by agent i. The vector field $v_i|_p$ associated with the flow Ψ_i is obtained by differentiation and the third identity in (15). This vector field, applied at p, is proportional to:

$$v_i|_p = \left[v_{i,n_1}, \left[v_{i,n_2}, v_{i,n_3}\right]\right] \tag{18}$$

Where [.,.] is the *Lie bracket* defined as follows: Let v, w, be derivations on M and let f be any real valued, smooth function $f : M \rightarrow R$ then the Lie bracket of v, w is the derivation defined by: $[v,w](f) = v(w(f)) - w(v(f))$ [11].

The composite action $v_i|_p$ generated by the ith agent to control the process is a composition of the form of (11). Moreover, from a version of the Chattering lemma and duality [12], we can show that this action can be expressed as a <u>linear combination</u> of the primitive actions available to the agent:

$$\left[v_{i,n_1}, \left[v_{i,n_2}, v_{i,n_3}\right]\right] = \sum_j \gamma_j^i(\alpha)\, v_{i,j}$$
$$\sum_j \gamma_j^i(\alpha) = 1 \tag{19}$$

with the coefficients γ_j^i determined by the fraction of time $\alpha_{i,j}(p,t)$ that each primitive action $v_{i,j}$ is used by agent i.

The effect of the field defined by the composite action $v_i|_p$ on any smooth function (equivalent function class) is computed by expanding the right hand side of (10) in a Taylor series (Lie-Taylor [11]). In particular, we can express the evolution of the unsatisfied demand C_i^u over the interval $[t, t+\Delta)$, starting at point p by:

$$C_i^u(t+\Delta, p'') = C_i^u\left(t, \Psi_i(t+\Delta, p)\right) \tag{20}$$

Expanding the right-hand side of (20) in a Lie Taylor series around (p, t), we obtain,

$$C_i^u(t+\Delta, p'') = \sum_j \frac{\left(v_i|_p\left(C_i^u(p,t)\right)\right)^j \cdot \Delta^j}{j!}$$

with

$$\left(v_i|_p(\cdot)\right)^j = v_i|_p\left(\left(v_i|_p\right)^{j-1}(\cdot)\right) \tag{21}$$

and

$$\left(v_i|_p\right)^0(\cdot) = \text{Identity operator}$$

In general, the right side of (14) will have countable, non-zero, number of terms. In our case, this series will have finitely many non zero terms. This is so, because in computing powers of derivations (i.e., limits of differences) we need to distinguish among different neighboring points. In our formulation of the topology of M, this can only be imposed by the clausal information. Since each agent's knowledge base has only finitely many clauses, there is a term in the expansion of the series in which the power of the derivation goes to zero.

2.2 Desired qualitative and quantitative closeness constraints of the closed-loop system

Detailed models of enterprise-wide processes of the kind mentioned above necessarily include both qualitative and quantitative constraints. For example consider the problem of integrating high-level scheduling of manufacturing production of a factory and low-level execution of manufacturing processes within a work cell of the factory:

- *Qualitative characteristics and logical constraints:* The scheduler for a factory must produce the schedule by evaluating throughput, tardiness, work-in-progress, machine wear, and other metrics. Each of these input-output criteria can be measured as a number. These numbers are global criteria (apply to the factory as a whole), and, since the criteria are not associated with a particular step in the manufacturing process, treat the factory as a black box (i.e. to be analyzed by input-output considerations). Other metrics are associated with quality of product which may be based on appearance, density, texture, thickness or some other variable which can be sensed, approximated by an allowable range of values, and used to create an event-based switching function for altering production quantity and flow through the factory.
- *Quantitative characteristics and continuum constraints:* Contrast this with the problem of deciding what to do next at an individual work cell. The logical decisions discussed above, normally captured as events occurring at instants in time, above must be compatible with the laws of physics which govern the continuous operation of motors, conveyor belts, sensors, actuators and so forth.

Further consider the problem of centralized control of the factory floor. If the factory is equipped with a single supercomputer and a very fast, unfailing sensor environment, then one program can be tasked with deciding for every work cell what it should do next, in complete detail. If communication lines to that computer fail, the factory stops. If its sensors lie, the factory may not stop, which can be worse. If simultaneous events overload the supercomputer, everyone waits. If small amounts of another product need to be produced between production runs of the current product, the plant must be reconfigured.

These considerations argue for a multiple-agent approach with coordination among work cells to achieve reasonable performance. While a high degree of autonomy among agents is needed to support a "divide-and-conquer" approach to meeting the complexity of producing workable plant schedules, it is possible for one controller to degrade factory performance by shuffling parts between two orders in a looping fashion, causing downstream workcells undue overhead in reconfiguring. Distribution of control is needed, so that work cells can make autonomous decisions, using advice from neighboring work cells, and performance criteria (goals) from a factory-wide goal-setting agent. Factory plans are made by considering certain constraints, normally in the context of a nominal manufacturing scenario. However, factory execution occurs in the context of actual decisions and events, which can deviate from the nominal scenario. In a general sense this is true for any application that spans the boundary between planning, where limited experimentation can be conducted, and situated activity, which has a much larger set of possible outcomes. The concept of an agent must appear on both sides of this boundary.

Multiple-agent, declarative control provides a path to substantially contribute to analysis and resolution of factory planning, scheduling and control.

2.3 Generation of programs (automata) to force closed-loop compliance with system constraints.

2.3.1 Bellman's optimality principle: A state-space oriented assertion of Bellman's optimality principle is that an optimal policy has the property that whatever the initial state and the initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision. For hybrid systems, application of this principle leads to the Hamilton, Jacobi, Bellman (HJB) relaxed variational form [37]:

$$V_i(Y,\tau) = \inf_{\alpha_i} \int_{\tau} L_i(\Psi_i(\tau,\ Y), v_i\big|_p(G_i(\tau,\ p))) \cdot\ d\alpha_i(p,\ d\tau)$$

$$\frac{\partial V_i}{\partial \tau} = \inf_{\alpha_i} H(Y,\ \frac{\partial V_i}{\partial Y},\ \alpha_i) \qquad (22)$$

$$Y(t) = p$$

$$\tau \in [t,\ t+\ \Delta)$$

Where **p** is a point in the carrier manifold, V_i are tangent vectors at each point and a_i are coefficients of the tangent vectors. Closed form solutions for this equation were previously only known for linear systems with quadratic costs where the solution of the HJB equation is achieved by solving the associated Riccatti equation.

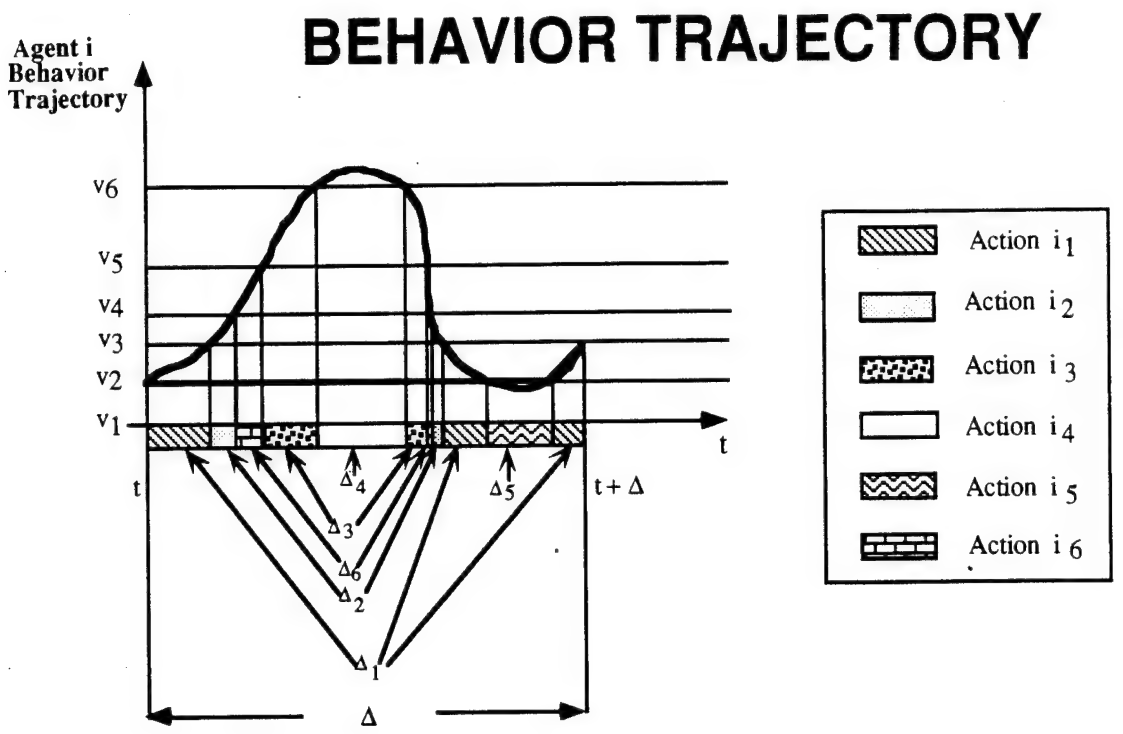


Figure 5. Agent i behavior as a chattering combination of infinitesimal actions

2.3.2 Kohn-Nerode-James optimality result:

Consider vector fields in the tangent bundle of infinitesimal actions (see Figure 5) and the tangent at point x in the carrier manifold.

What happens when moving to point x + Dx?

In ordinary calculus we would use the directional derivative to answer the question. In the mathematics of manifolds, the Levi Civita theorem provides a mechanism for connecting the tangent at x to the tangent at x + Dx.

In the Kohn-Nerode formulation for hybrid systems, there is no distance metric but there is a concept of closeness based on the Kohn-Nerode definition of continuity for hybrid systems. The application of the definition of continuity leads to the construction of the HJB equation and its solution as the chattering combination of infinitesimal control actions (Figure 5).

The appropriate interpretation of this solution is that the coefficients of the tangent vectors satisfy the Levi Civita continuation equation (affine connection) from x to x + Dx and coincides with the HJB equation from x to x + Dx.

The revolutionary consequence of this result is that the Bellman Optimality Principle is the consequence of continuation in the hybrid systems formulation. The effect is that instead of having the previous limited result for linear systems that *if a solution exists*, then the LQ or H^{∞} result is exact, we can now use infinitesimal actions to both define the choices available, and then construct the HJB equation to be solved, secure in the knowledge that the T0 topology of hybrid systems tends to the Hausdorf topology in the limit [see 37 for a proof]. The result is summarized below:

We note that both actions and disturbances are transformations on points in the carrier manifold. Each infinitesimal action or disturbance is represented as a derivation on M as discussed above. We construct an inference automaton which is the epsilon-optimal hybrid controller (D approximation). The evolution equations are:

$$\begin{aligned}
&\mathbf{q}_{t+\Delta} = \delta\left(\mathbf{q}_t, \omega_{t,t+\Delta}\right) && \text{neighborhood transition}\\
&\omega_{t,t+\Delta} && \text{solution of current equational terms}\\
&\mathbf{v}\big|_{\mathbf{q}}(\ ) = \beta(\mathbf{q}_t) && \text{infinitesimal action}\\
&\mathbf{u}_{t+\Delta} = \mathbf{EXP}\left(\Delta \bullet \mathbf{v}\big|_{\mathbf{q}}\right)\left(\mathbf{H}(\mathbf{X}, \mathbf{g})\right) && \text{current control law}
\end{aligned} \tag{23}$$

The logic equations are:

$$\begin{aligned}
&\mathbf{Y}_t = \mathbf{E}(\mathbf{q}_t) \bullet \mathbf{Y}_t + \mathbf{K}(\mathbf{q}_t) && \text{Kleene - Schutzenberger Equation}\\
&&& \text{resolution of current relations}\\
&\omega_{t,t+\Delta} = \mathbf{S}(\mathbf{Y}_t) && \text{Selector function}
\end{aligned} \tag{24}$$

3. Discussion of MAHCA relative to other enterprise-wide architectures:

3.1. An overview of MAHCA : MAHCA supports elimination of the current *ad hoc* approach for integration and evolution of heterogeneous components of large-scale systems by providing a rigorous methodology, reference architecture, and (preliminary) tools for incremental construction of large-scale systems. MAHCA applies a *hierarchical organization* of enterprise-wide infrastructure (similar to the ISO layered network model), and a logic distribution of *control agents* controlling these processes.

3.1.1 Hierarchical organization of enterprise-wide applications:

For the manufacturing case, the need is to organize the infrastructure for factory planning, scheduling and control (PSC). MAHCA uses a layered architecture (Figure 6) to implement the control of enterprise-wide processes.

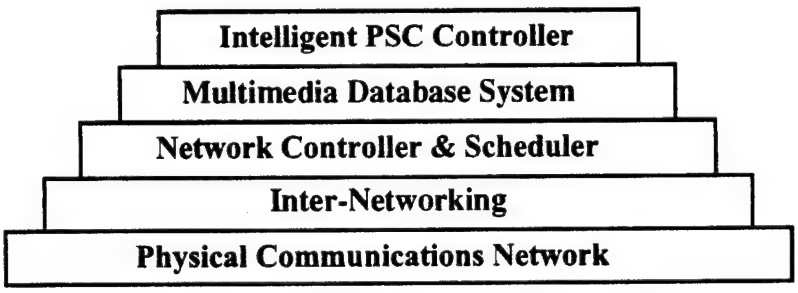


Figure 6. Hierarchical Organization of Enterprise-Wide Infrastructure

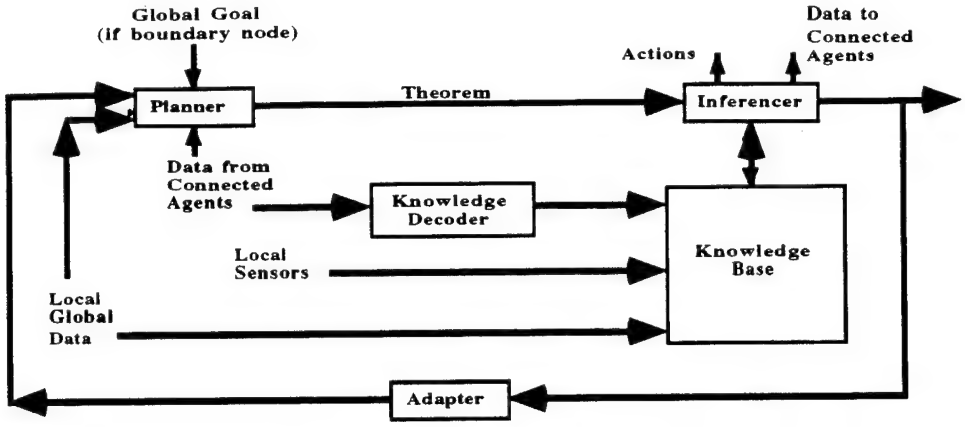


Figure 7. Control Agent

3.1.2. Logic distribution of agents:

The PSC agent network is composed of a (varying) number of devices called *Control Agents* (Figure 7) implicitly coordinated through a variable (over time) configuration *Network of Control Agents* (Figures 8) implemented through the hierarchy of Figure 6. In order to accommodate the reliable provision of control of the flow of products and partial products through the manufacturing workcells, we intend that the network be controlled by a network of autonomous agents, the PSC agent network. The fundamental (global) goal of the PSC network is to resolve unsatisfied demand between producers and consumers. *Conservation of unsatisfied demand* is the primary logic condition for synchronization of the agents in the PSC network. Control of the network is *implicit without umpire* since synchronization is effected through sharing of constraints between agents. The inter-agent communication network's main function is to transfer inter-agent constraints among agents according to a protocol written in equational Horn clause language. These constraints include application dependent data and, most importantly, inter-agent synchronization. The inter-agent synchronization strategy is very simple. An agent (Figure 7) is synchronous with respect to the network (Figure 8) if its inter-agent constraint multiplier is continuous with respect to the current instance of its active knowledge. Since the equational Horn clause format allows for the effective test of continuity (which is implicitly carried out by the inferencer in the Knowledge Decoder), a failure in the Knowledge Decoder theorem results in a corrective action toward re-establishing continuity with respect to the topology defined by the current instance of the knowledge base (see [7, 8, 9, 10]). As mentioned in the introduction, structural adaptation is accomplished by the modification of the logic clauses, according to a set of modification rules, or by the creation or deletion of agents in the network. A significant consequence of this structural adaptation approach is the ability to support on-line modification of a hierarchical partitioning of the manufacturing process such that, from the highest level of abstraction, decomposition of manufacturing tasks into a *continuum hierarchy* is supported.

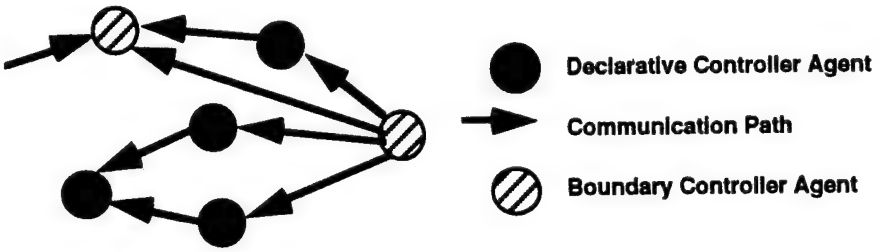


Figure 8. Network of Cooperating Control Agents

The specification of the geometry of the network, as a function of time, is dictated primarily by global observability. By global observability we mean closure of the knowledge of the system as whole relative to the scope the systems reactivity.

3.1.3 Architecture capabilities:

The multiple agent hybrid declarative control architecture exhibits several key capabilities, for the implementation of high-safety, high-assurance systems. We list some of them next.

Reactivity: The theorem proving function of each agent on the architecture operates according to a first principles feedback paradigm. This property extends the feedback principles to the decision levels in the manufacturing process, which allows for a status-dependent tuning of their actions.

Adaptivity: The knowledge base of each agent is open and modifiable by sensory data. Theorem failure triggers tuning and corrective action. This property maintains the validity of the knowledge base of the agents and is the basic mechanism in the feedback paradigm discussed above.

Distributive with Coordination: The theorem proving is carried out distributively over the agents. The coordination scheme is implicit without umpire. This property is central for ensuring that although each agent controls a local aspect of an enterprise, the global logic integrity of the system is maintained over time.

Dynamic Hierarchization: The architecture can operate simultaneously at different levels of abstraction. The Knowledge elements that characterize an enterprise are naturally distributed over many levels of abstraction. The characteristics of these levels are a function of process status. Therefore, in order for the agent controllers to maximize the use of the available knowledge its hierarchization must be tuned to current status.

Figure of Merit: The behavior of the closed loop distributive system is determined by proving that there exists a command trajectory that minimizes a goal functional. The selection of this functional depends both on short and long term agent objectives which again depend on current status.

Real-Time: Constraints for real-time performance are explicit and part of the knowledge base. This is important because in a complex real-time environment, the timing of normal or unexpected events is strongly correlated with the status of the process and therefore the structure of the real-time constraints cannot be fully instantiated at design time. MAHCA supports adaptive optimization of large-scale sytems through on-line solution of a nonlinear HJB equation which is then used for program generation.

The central mechanism for providing these capabilities is an on-line restrictive mechanical theorem prover within each agent . The architecture consists of a Knowledge Base which stores the goal for the agent, system constraints, inputs and inference operations. A Planner generates the theorem which represents goal. For some agents, this goal will govern the behavior local to that agent. For other agents, the goal will also include behaviors global to the system. The Inferencer proves the theorem. If the theorem is true, control actions, computed during inferencing, are issued to the plant. If the theorem is false, an Adapter processes the failed terms in the theorem for replacement or modification. Data from other agents is provided to the Planner for incorporation as constraints into the theorem and passes through a Knowledge Decoder for entry into the Knowledge Base. We provide the following summary of results for the multiple-agent hybrid control architecture (MAHCA)

1. MAHCA supports construction of a precise statement of the enterprise control problem in terms of multiple agent hybrid declarative control. Our approach characterizes the problem

via a knowledge base of equational rules that describes the dynamics, constraints and requirements of the processes being controlled (channels, switching modes, customer characteristics, scheduling and planning strategies, etc.).

2. We have developed a canonical representation of interacting networks of controllers. Given a connectivity graph with N nodes (controllers) and the corresponding agent's knowledge bases, a network of 2N agents can be constructed with the same input-output characteristics, so that each agent interacts only with another (equivalent) companion agent, whose knowledge base is an abstraction of the knowledge in the network as viewed by the agent. Thus, in general, the multiple-agent controller for any network configuration is reduced to a set of agent pairs comprised of an agent and its companion.
3. One agent of each pair, the Companion Agent, coordinates with other agent pairs across the network. The proof carried out by the Companion Agent generates, as a side effect, coordination rules that define what and how often to communicate with other agents. The coordination rules also define what the controller needs from the agent network to satisfy control requirements of its physical layer.
4. Our approach is based on a canonical procedure to prove a special class of existentially quantified theorems whose statement characterizes the desired behavior of a manufacturing PSC process, a state trajectory of the process that satisfies requirements in which the agent is involved. The procedure proves the theorem, with respect to the current knowledge base status, by constructing and executing on-line a finite state machine called the "proof automaton." The output function of this automaton generates the control actions the agent uses to control the aspects of the process in which it is involved.
5. In the Multiple-Agent Hybrid Control Architecture (MAHCA), the agent execution schema has linear complexity in the number of relational terms and the number of variables.
6. The Bellman Optimality Principle is the consequence of continuation in the hybrid systems formulation. The effect is that instead of having the previous limited result for linear systems that, if a solution exists, then the LQ result is exact, we can now use infinitesimals to both define the choices available, and then construct the HJB equation to be solved, secure in the knowledge that the T0 topology of hybrid systems tends to the Hausdorf topology in the limit.

This is the basis for generating procedures to interface heterogeneous components of the manufacturing process, and for our belief that the resulting architecture will support incremental expansion of new components with greatly reduced requirements for expensive experimentation validation. We do not expect to fully eliminate the need for experimentation because the degree of "trust" in the newly composed architecture will depend on the rules for composition of the components. However, to the degree that the composition rules are correct, the methodology will be a formally correct composition of the components, the focus of the verification and validation effort will be raised to the component level and the results will be reusable across the confederation of components.

How the existing hybrid systems unification mechanism works: We have been developing single-agent and multiple-agent demonstrations of extracting global and local control programs. The technology and the demonstrations have been developed under sponsorship from the Army Research Office (ARO), the Ballistic Missile Defense Office, Army Armaments Research, Development and Engineering Center (ARDEC) and Department of Defense Advanced Research Projects Agency (ARPA) [7, 9, 21].The sequence of steps leading to *generation* of programs which comply with the *current* specifications and parameter values are:

1. The original problem is reformulated as a calculus of variations problem on a carrier manifold of system states. The carrier manifold is the combined simulation model of the network and the simulation models at the nodes, a manifold on which the (evolution of) state trajectories occur. We are to find control functions of the *state* of the system for the global and local problems which minimize a non-negative cost function on state trajectories whose minimization perfectly achieves all the required goals of the ADS.

2. One replaces the variational problem with a convex problem by convexifying, with respect to u, the state rate, the Lagrangian $L(x,u)$ which is being minimized. The convexified problem has a solution which is a measure-valued (weak, or L. C. Young) solution to the original problem. This is a chattering control which chatters appropriately between local minima of the original problem so as to achieve close to the global minimum of the original problem. This solution, however, is only abstract and gives local and global control functions of *time*.
3. To get control functions of *state* instead, we convert the convexified problem to the appropriate Hamilton-Jacobi-Bellman equation form. An 'e-solution' of this equation for the appropriate boundary conditions, gives valid infinitesimal transformations on the state space representing the generators of feedback controls, i.e., control functions of state, not time, the solutions dual to the original ones. The weak solution obtained is approximated to by global and local "chattering control" programs.
4. The controls that are possible at a given state are a cone in the tangent plane, and move with the tangent plane. If one follows the optimal control as one moves in state, the near optimal controls needed are algebraically represented by the Christoffel symbols of an affine connection, a recent result of Kohn-Nerode-James [37]. The Christoffel symbol representation gives the real-time computation of the global and local automata, or control programs, needed to govern the communications network and the approximations at nodes in order to meet the prescribed goal. The global program takes responsibility for message passing between nodes, the local programs take responsibility for local updates in real time. Required dynamics of the global system is achieved without central control (umpire) for the distributed system through enforcing global continuity conditions at each node.

To our knowledge, the Kohn-Nerode approach to hybrid systems is the only theory to both (1) provide a solid mathematical foundation to unify logical and evolution models and (2) be computationally feasible. The methodology, while promising, is very new and extensive research is required to understand both the most effective ways to construct the unified models and to build the necessary interfaces to existing systems. The rapid acceptance of the theory by major research institutions attests to its relevance to fundamental issues in several disciplines.

3.2 Comparison of MAHCA to other architectures:

3.2.1 Discussion of NASREM:

For over twelve years the National Institute of Standards and Technology (NIST) has sponsored a series of improvements in achieving a reference architecture for distributed intelligent control systems. The hierarchical control system developed for the Automated Manufacturing Research Facility at the National Bureau of standards [1] was developed as an architecture for machine shop control and subsequently used for control of multiple autonomous undersea vehicles and battle management [3, page 1]. The NASA/NBS Standard Reference Model for Telerobotic Control System Architecture (NASREM) (see [3,4] and Figure 9) has been a valuable reference architecture for a variety of distributed, real-time systems.

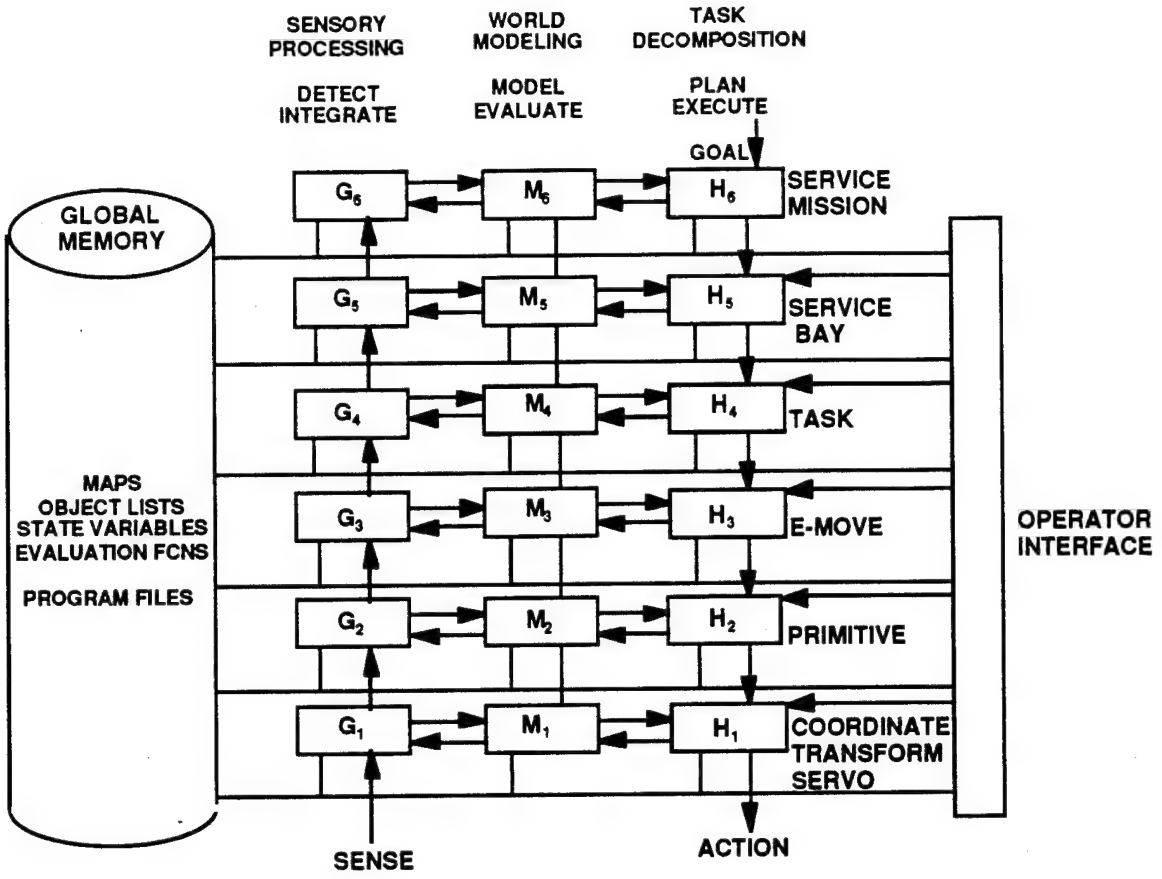


Figure 9. NASREM: A hierarchical reference architecture for control of telerobots

NASREM was created to be used as a guideline for the development of the control system architecture of the Flight Telerobotic Servicer for the Space Station. NASREM is currently the reference architecture for a number of research activities, including experiments in mobile robotics at the US Army Armaments Research, Development and Engineering Center (ARDEC) and in command and control architectures for the Advanced Research Projects Agency (ARPA). Furthermore, NASREM was conceived to support life-cycle activities in construction and maintenance of large-scale, real-time systems [3]:

> The NASREM telerobot control system architecture defines a set of standard modules and interfaces which facilitate software design, development, validation, and test, and make possible the integration of telerobotics software from a wide variety of sources. Standard interfaces also provide the software hooks necessary to incrementally upgrade future Flight Telerobot Systems as new capabilities develop in computer science, robotics, and autonomous system control.

However, a strict adherence to a layered architecture like the NASREM architecture is known to have occasional problems with stability for lower-level control components and with logical inconsistency for higher-level planning components, leading to a need for conducting extensive experiments to discover and compensate for failure modes of the system. The multiple agent hybrid control architecture (MAHCA) provides the flexible structure and mathematical rigor required to resolve these known problems with stability and logical inconsistency while enhancing the existing NASREM capabilities for:

- incremental construction,

- integration of heterogeneous components,
- collaboration among diverse disciplines (multiple views),
- verification, validation and accreditation (commissioning), and
- accommodation of new technologies.

The problem with stability for control systems is related to the issue of knowing when to make a decision (execute a control action) for affecting a controlled value (desired result) at some time in the future. At the lowest layers in the hierarchy (See Figure 10), the time required to execute a commanded action (control law) is normally quite small (on the order of microseconds or milliseconds) while at the highest layers in a complex system, it might be hours or days.

Sense	Model	Act
Layer n		
Layer n - 1		
Layer 2		
Layer 1		

Figure 10. Layered Architecture with fixed hierarchy of layers

The normal implementation decision is to partition the operating conditions of the system into well-understood *modes* for which adequate low-level controls are designed. Higher-level decisions, which occur at much slower time scales, then occasionally initiate switching between operating modes (such as start-up, shut-down or normal-operation). This is reflected in the order-of-magnitude differences in output rates between each layer of NASREM (see Table 1) and corresponding longer planning horizons and average replanning intervals.

For small-scale systems and for systems which change very slowly over time, this approach has worked well for decades, since an *ad hoc* mixture of components can be experimentally tested with a high-enough assurance that catastrophic failure modes have been discovered and adequately addressed. However, for larger-scale distributed systems and for systems which change more rapidly, there is not enough time and/or resources to adequately execute sufficient experiments for verification, validation and accreditation (commissioning) of the system.

Layer	Average rate of change in output	Average replanning interval	Planning horizon

Servo	1000 Hz	2 milliseconds	15 milliseconds
Primitive	100 Hz	30 milliseconds	300 milliseconds
E-Move	10 Hz	200 milliseconds	2 seconds
Object/Task	1 Hz	3 seconds	30 seconds
Service Bay	0.1 Hz	1 second	> 10 minutes
Mission	0.01 Hz	6 minutes	> 1 hour

Table 1. NASREM timing relationships between and within hierarchical layers

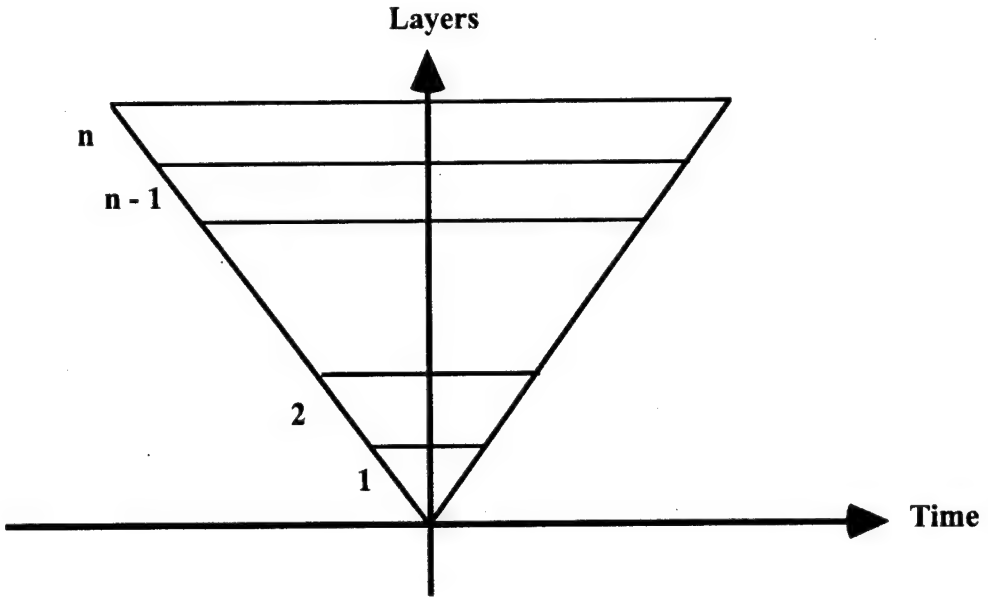


Figure 11. Time delay associated with fixed hierarchy

This has serious implications for system stability since the difficulty occurs when logical decisions at higher levels are allowed to affect parameters which determine stability of low-level components. In that case, the time delay associated with different layers in the hierarchy (See Figure 11) means that accurate predictions of system behavior are needed to be made farther in the past for current actions to be correct. For linear models it is easy to see that sufficient time lag can be easily introduced to cause phase margins for system stability to be exceeded. These conflicts can be predicted by closed-form solution of the linear models and adequately addressed in the system implementation. However, since linear models only occur for small perturbations around equilibrium operating conditions for different modes, it is not possible to accurately predict system behavior without conducting sufficient experiments for an allowable range of parameter values. A similar problem is present in the case of lower-level events causing changes in logical consistency of higher-level plans. What is needed is a rigorous methodology for enabling the occasional event at higher levels in a hierarchy to directly affect (without time lag) the actions(s) at an appropriate lower level and for the occasional event which occur at lower levels in the hierarchy to directly affect (without time lag) the plans and decisions at the appropriate higher level. Hybrid systems theory provides that methodology and we have designed MAHCA as a reference architecture based on the methodology.

MAHCA supports establishment of a distributed team of cooperating reasoning agents connected in a possibly time-varying logic network. The domain of expertise of each agent is represented in its stored knowledge and the knowledge that flows to it from its sensory inputs and other agents. This knowledge encodes information of the characteristics of a particular manufacturing process in execution as it *relates* to desired *quality of service* or *manufacturing performance metric* behavior and also to a possibly time varying **goal**. The function of each agent is to **infer** from the stored knowledge, and the actual data (sensory information and partial resolutions coming from the other agents) a combined resolution that instantiates a *model*, resolving the goal. This model and the extraction process satisfy pre-established QOS or performance metrics. Prototype software to implement the extraction process currently exists and projects to demonstrate both single and multiple-agent applications of the extraction process are being supported by the Army and the Defense Advanced Research Projects Agency. MAHCA has been shown to be capable of *synchronizing events and continuum values* (maintaining stable operation) across a broad range of time scales and we have developed preliminary concepts for *synchronizing semantic agreement* between plans and implementations (maintaining consistent views) across a broad range of disciplines.

Lie algebra results concerning infinitesimal operators on smooth functions allow us to consider all the standard evolution models of differential operators and DEDS evolution models. We embed logical models in continuous models in order to construct automata which comply with logical and continuum constraints. The data flow model of points in the carrier manifold (corresponding to single-agent control implementation) is given in Figure 12. We assert and emphasize here that for systems which meet the conditions for creation of a hybrid system state, the revolutionary nature of our approach has two benefits:

- Creation of a unified mathematical foundation for analysis and synthesis of models which for decades have been treated separately, and
- Creation of a rigorous process for incremental expansion of trusted systems which must comply with stringent safety and performance constraints.

3.2.2 Discussion of Advanced Research Projects Agency (ARPA) Domain-Specific Software Architectures (DSSA) program:

The architectural ideas upon which MAHCA is based are derived in large part from the excellent results available from previous DSSA efforts to begin programming by components. Indeed, development of MAHCA has been partially funded by the ARPA DSSA program. The overall objectives of the DSSA program are stated in [39]. These ideas include construction of component-based domain models, creation of reference architectures, use of architecture description languages, and cost reduction through software reuse.

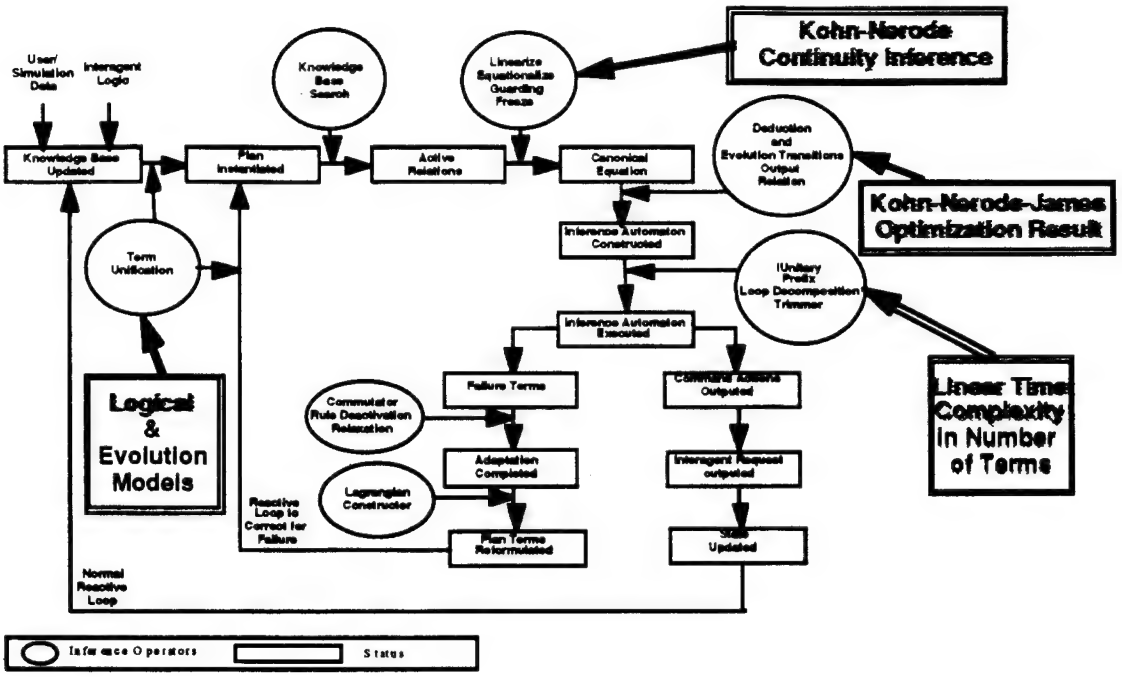


Figure 12. Data Flow Model of Points in the Carrier Manifold

As developed early in the DSSA program, one definition of a DSSA is that it is: an assemblage of software components,

- specialized for a particular type of task (domain),
- generalized for effective use across that domain,
- composed in a standardized structure (topology) effective for building successful applications in the domain.

Component-Based Domain Models: The DSSA program defines a domain model as the terminology and semantics characterizing elements and relationships in a domain. The domain model defines the terms used to express requirements and evaluate systems. Models used to build DSSAs suffer from the fact that there is a split in the enabling fundamental technologies used to create and compose the models. As indicated in the DSSA Guidelines report:

> Domain modeling needs to address both the system and its environment. Often a system is composed of many ingredients, only a fraction of these being software. The domain model must address all aspects of the system performance in the environment that will affect the requirements for or constraints on the software. An excellent domain model should allow us to determine whether a system as designed will meet the objectives of the system as required. Synthetic environments which simulate the designed system in its operational context can provide this type of capability [44].

A basic motivation behind the DSSA program is to lower costs of integration of future systems by providing a structure (the architecture) and a process (guidelines for developing and applying the architecture) within a narrow application area (the domain) in order to reduce the costs of integration of future systems. While the idea of the DSSA guidelines to have an all-inclusive

model is a desirable goal, the diversity of models used to capture different portions of environments is a barrier to cost-effective achievement of combined models (See Figure 13 [43]).

The split in models carries beyond those usually associated with real-time control and is a pervasive situation for DoD systems. The Defense Science Board task force focused on tools and technologies for Distributed Interactive Simulation (DIS). Figure 13 shows the dependence of each layer of DIS implementation on two foundational kinds of models: (1) human behavior representation models of cognitive behavior and (2) environmental representation models of the physical environment.

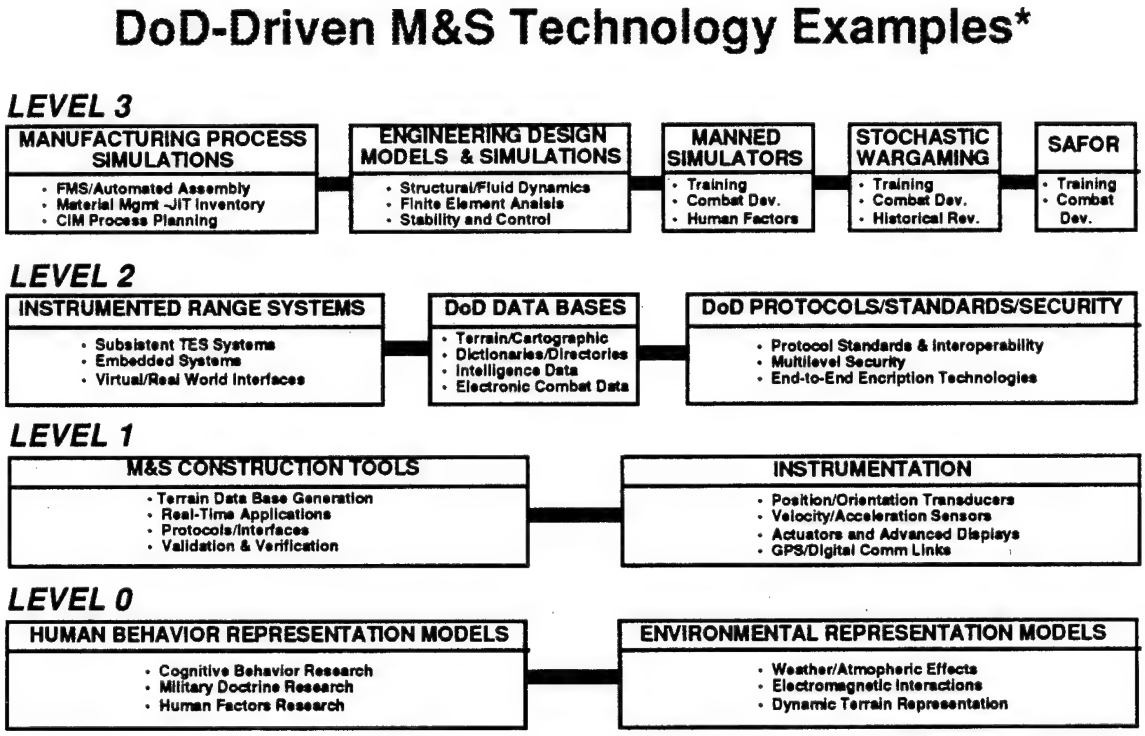


Figure 13. Examples of Diversity of Models

These models use fundamentally different mathematical tools. Cognitive models are built using linguistic (in the Computer Science sense) tools which depend on the set-based mathematics of algebraic topology. Models of the physical environment are built using simulation tools which support experiments with compositions of set-based, linguistic (logical) models and continuum-based models which depend on the mathematics of differential operators (normed vector spaces for systems of linear differential equations). Experiments are necessary to determine the behavior of the composition of models for safety, reliability and performance constraints. These models are subsequently used to build progressively more complex systems at higher levels of integration until ultimately they are used in requirements definition, prototyping, program planning, design and manufacturing, training and readiness, and test and evaluation.

We are convinced that the key contribution of hybrid systems theory to component-based programming is the creation of domain models which can be more easily integrated. Easing the integration process is a key contribution because the current process is ***fundamentally limited*** by the fact that the approach for integration of diverse models is to apply engineering experience and heuristics to combine diverse models and then to exhaustively experiment with alternative parameter values or structural adaptations until satisfied with the result. While progressively more effective methods for disciplining the process and checking on the results have been developed, the experimental nature of the approach has not changed substantially in several decades. Establishing a more disciplined component-based software engineering process and creating tools and organizations to implement that process will substantially lower the costs of integration of complex systems. However, hybrid systems theory is a path to provide the software engineering process with a fundamentally new means to formally define and construct compositions of diverse models, reducing or eliminating the need for exhaustive experimentation. This situation is especially promising for integration of real-time systems into larger information systems.

Software Reuse: One of the useful results from the Domain-Specific Software Architectures (DSSA) program has been a high-level concept for the relationships between domain management, domain engineering, the problem space of domain engineering and the solution space of application development (see Figure 14).

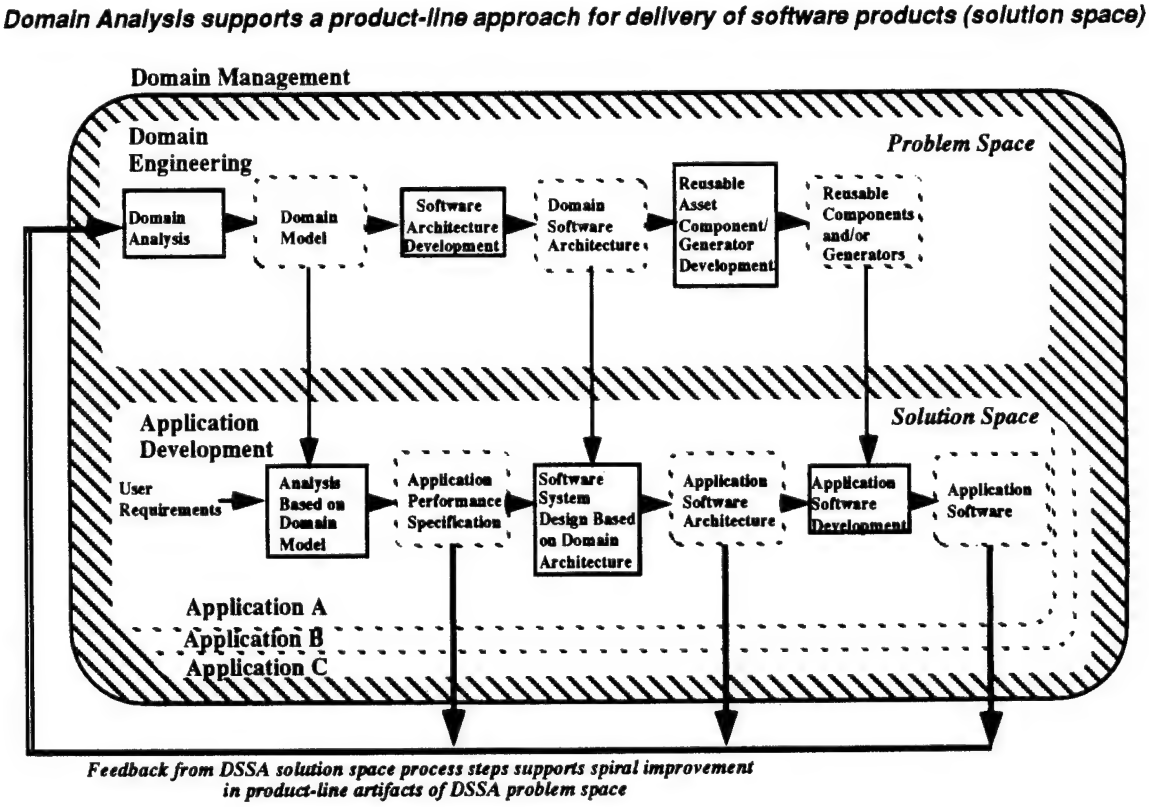


Figure 14. Use of Domain Analysis Products to Support Application Development

The *problem space* contains information delineating what is to be done to resolve problems encountered in the domain without defining how to do it. The *how* is the purview of application

development. Domain Engineering is the conduct of activities related to understanding the problem space and include:

- *Domain Analysis:* Identification, analysis, and development of a domain model.
- *Software Architecture Development:* Development of the Domain-Specific Software Architecture (Reference Architecture)
- *Composition of Reusable Components or Component Generation*

Products of Domain Engineering of the problem space include:

- *Domain Model:* Identifies entity classes, inter-entity relationships, and operations on entities, which are common to most systems in a given specific domain.
- *Domain-Specific Software Architecture (Reference Architecture):* A structural model representing a high-level design packaging of functions, their relations and control, for a family of related systems, with underlying reusable components designed to work together to support the implementation of applications in the domain.
- *Domain Implementation (Reusable Objects):* To construct robust, reliable source code which can be used to configure and adapt DSSAs to meet specific system requirements (reusable components or component generators).

While domain engineering concentrates on the development of the above products, the utility of the products of domain analysis rests upon the extent to which they are used in the application development process. These activities occur in the Solution Space: and comprise the delineation of how to resolve a problem (application development):

- *Analysis* of user requirements based on the domain model,
- *Software System Design* based on the domain reference architecture, and
- *Application Software Development* based on composition of reusable components or component generation.

Products of the application development process in the solution space include:

- *Application Performance Specification:* Developed using the Domain Model
- *Application Software Architecture:* The instantiation of the DSSA (Reference Architecture) for the problem at hand.
- *Application Software:* Application of Domain Implementation products

Thus, domain models are used in the requirements analysis phase of application development to produce application performance specifications. Likewise, DSSAs (reference architectures) are used in the design phase to develop application specific architectures and reusable components are used in the development phase to produce the target application. Domain management is the responsibility of the functional proponent for the domain. Figure 14 is a modification of a slide used by John Leary at a DSSA meeting and summarizes many of these ideas..

It is important to note that the domain engineering and application engineering processes are seen as repetitive with incremental improvement being achieved through progressively more accurate prototypes. Concurrent feedback supports spiral development and the Domain Engineer and Application Engineer coexist in an environment supported by a reuse repository or library. The levels and types of support provided can vary from a customized reuse environment supporting a vertical domain to a repository supporting many domains in a rather generic unintegrated manner. The Department of Defense has recently initiated an integration of several repositories. In addition, the Reuse Library Interoperability Group (RIG) has produced several documents to facilitate construction of open-architecture repositories. The RIG has also recently published a uniform data model for reuse libraries [45] and a report on measuring the interoperability of reuse libraries [46].

Reference Architectures: The DSSA program has defined a reference architecture as a software architecture for a family of applications in a domain. The program defines a software architecture as an abstract system specification consisting primarily of functional components described in terms of their behaviors and interfaces and component-component interconnections. The interconnections provide means by which components interact.

DSSA projects have developed reference architectures for avionics [40], vehicles [41], and guidance, navigation, and control [42]. We will next discuss architecture work in each of these areas .

3.2.2.1 Discussion of Teknowledge DSSA architecture: The Teknowledge approach to architecture development and application is to view the DSSA products in terms of three basic levels of systems where (1) the domain architect uses the Domain Development Environment to produce the reference architecture, components and development tools which comprise the Domain-Specific Application Development Environment to be used (2) by the application engineer to produce an architecture instantiation which is used (3) by the operator as the Application Execution Environment.

The Cimflex/Teknowledge team is integrating ABE, BB1 and their Requirements Manager to form an Application Development Support Environment (ADSE). Cimflex Teknowledge is working to assimilate intelligent/hybrid control into their Distributed Intelligent Control and Management (DICAM) architecture. The development of a partitioned hierarchy as a modified version of Albus' architecture for computer-controlled systems is seen as desirable. The sequence of development is seen as: representation, specification, compilation, and implementation. The hierarchical DICAM Reference Architecture is composed of (1) an information base and world model which uses a variety of knowledge representation techniques to capture information on the past, present and future of the different levels of the system and (2) a hierarchical network of semi-autonomous individual controllers. The Individual Controller Reference Architecture is shown in Figure 15.

Stanford University is working with Teknowledge to implement Intelligent Control From the Bottom Up. The position is that the higher-level decisions depend upon proper operation of the low-level controllers. Stanford has provided a discussion of the development of real-time controllers and the evolution of adaptive control into he field of intelligent control. Plans are for intelligent control to involve the on-line redesign of the control law based on an update of the plant model (on-line identification of system dynamics). The eventual result will be implementing intelligent real-time adaptive control within the DICAM architecture.

The Teknowledge-Stanford team has identified which portions of Figure 15 would perform the required functions for implementing an intelligent controller:

A real-time path from sensors, to input filters, plan executor, and effectors which reside in the Domain Controller,

System identification and control design functions which reside in the situation assessment, planner, assumption analyzer, local world model, and plan cache in the Domain Controller,

Identification and Control Design Assistants which reside in the Meta-Controller, and

Design Specifications and Modeling Information Database which are a component of the DICAM Information Base & World Model.

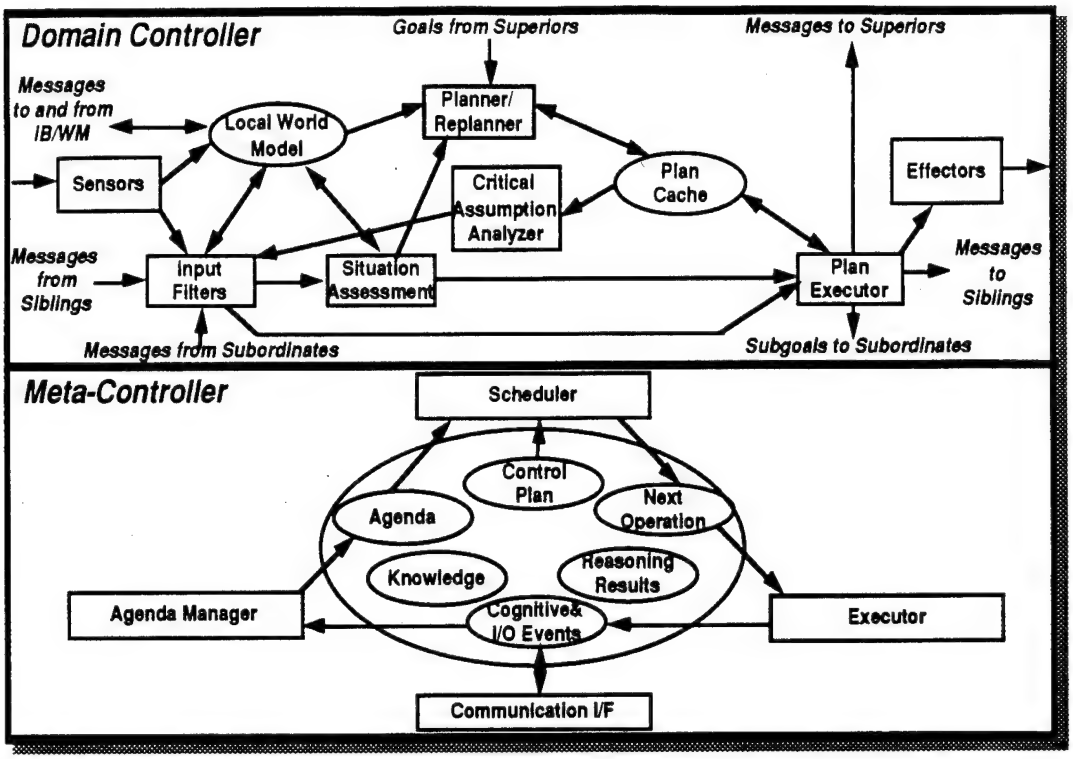


Figure 15. DICAM Individual Controller Reference Architecture - Domain Control and Meta-Control

A central feature of the environment being built by Cimflex/Teknowledge is the use of Knowledge-Based Design Advisors to provide advice, impose policies, and automate testing and design. It is expected that these can be effectively used for determining the most general requirements of the system as well as the most specialized. The DICAM architecture is based on the NASREM architecture and shares the problems of a fixed hierarchy of components for each implementation. Also, while it is expected that architecture-based reuse and generation of components will ease the integration process, the Teknowledge architecture relies on heuristics and experimentation to integrate logical models with continuum models. For large-scale high-safety, high-assurance systems, the experimentation approach for verification and validation does not have a path to success.

The individual controller reference architecture is being further specialized as a Generic Architecture for the Advanced Howitzer (see Figure 16). Initial portions of the howitzer architecture have been coded. Live demonstrations have been given of the use of the ADSE to capture the lower levels of the howitzer architecture and demonstrate features of the ADSE. One specific need being addressed in the howitzer architecture is support for construction of crew decision aids as well as integration of such higher-level functions with low-level continuous-time controllers. One of the crew decision aids being developed is the Reconnaissance, Selection, and Occupation of Position (RSOP). Such applications are expected to be developed using a process-based application development model which represents processes as networks of actions and results and mixed-initiative, policy-driven control of process execution, modification and enforcement.

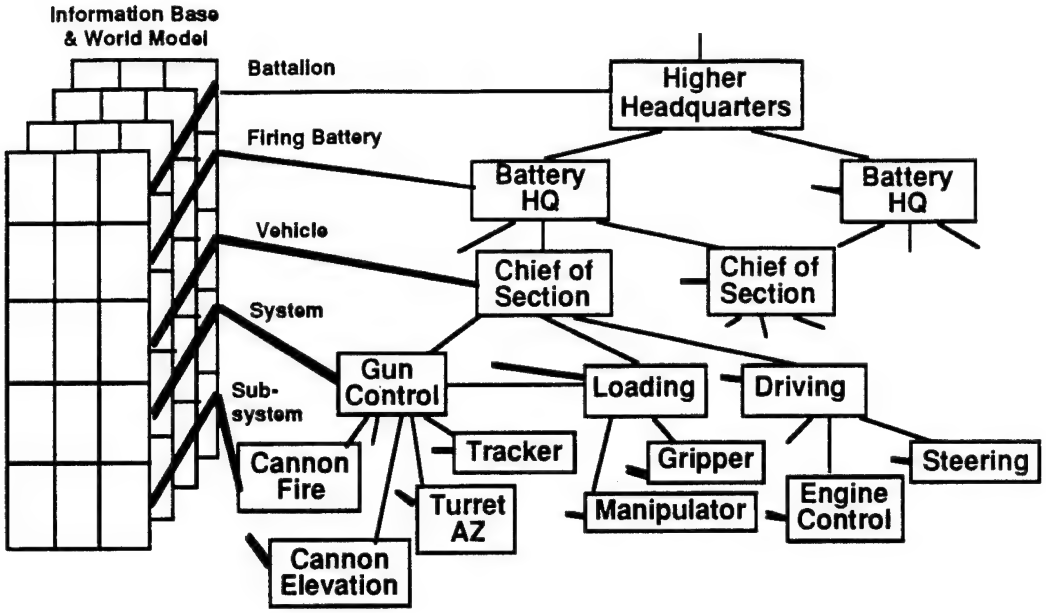


Figure 16. Specializing the Generic Architecture for the Advanced Howitzer

3.2.2.2 Discussion of Loral architecture:

The Loral effort is termed the DSSA Avionics Domain Architecture Generation Environment (ADAGE) project. The DSSA-ADAGE domain architecture development environment supports analysis and generation of avionics components using modules for Avionics Architecture:

- Profile generation
- Navigation
- Flight Director
- Control

The DSSA-ADAGE environment model is used to provide information on selected and derived data including:

- Data Source Objects (DSOs) are manipulated in the environment
- Logical Data Source Objects (LDSOs) combine multiple sensor inputs
- Derived data is computed from sensor values

The DSSA-ADAGE system configuration is determined at build time. Build time features include:

- Sensors for selection during operation
- Choice of a variety of filters
- Selection criteria for navigation modes and guidance modes
- Parameter values for filter constants, time constants,...

The Loral effort has focused on creation of a DSSA engineering process for creation of avionics software with an emphasis on generation of components which comply with specifications. Like the NASREM and DICAM architectures, the DSSA-ADAGE architecture shares the problems of a fixed hierarchy of components for each implementation. Also like the NASREM and DICAM architectures, while it is expected that architecture-based reuse and generation of components will ease the integration process, the Loral architecture relies on heuristics and experimentation to integrate logical models with continuum models.

3.2.2.3 Discussion of Honeywell architecture:

Honeywell views the DSSA program as resulting in a Guidance Navigation and Control (GNC) product line reference architecture which can be specialized into instances of a product line (e.g. an air-to-air missile GNC or an aircraft GNC). Honeywell views the DSSA Development process as a sequence of:

- Capture domain model (define problem space),
- Capture reference architecture (define solution space), and
- Reduce to product through iterative refinement of the problem space and solution space.

The information flow necessary to produce instances of the GNC architecture is shown in Figure 17.

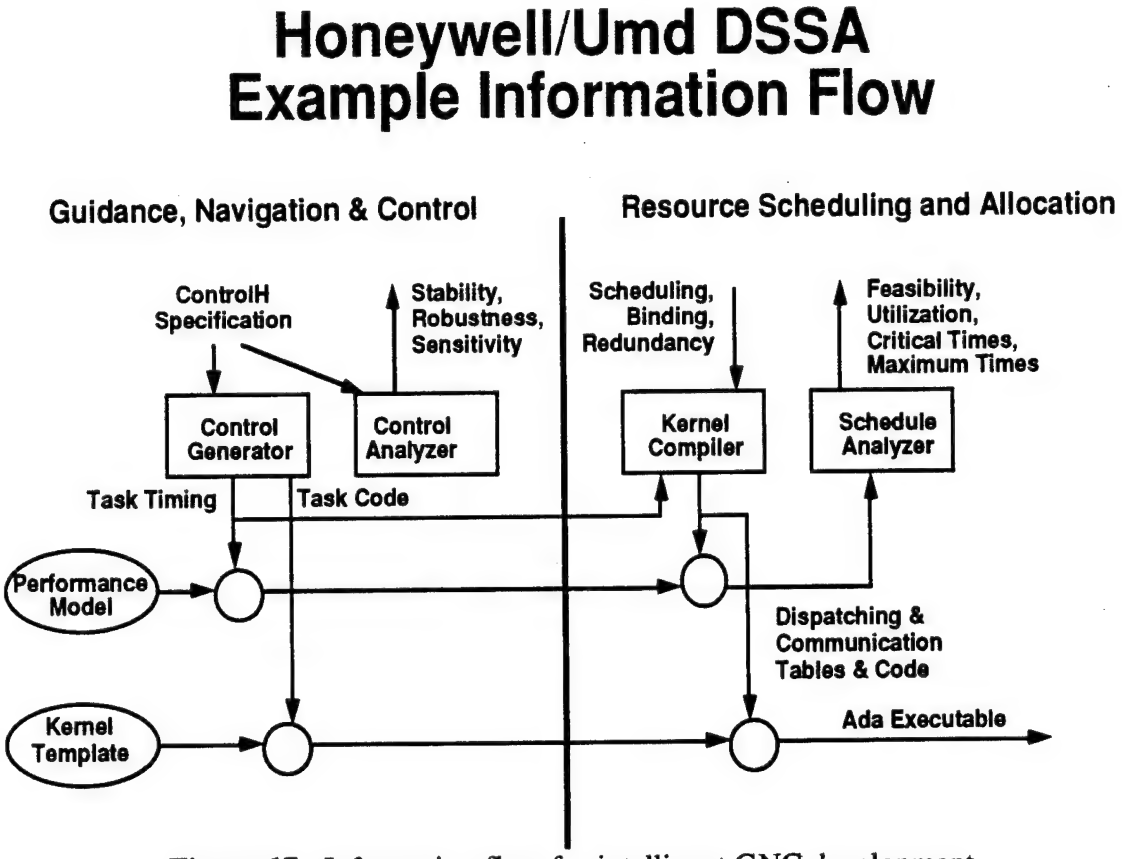


Figure 17. Information flow for intelligent GNC development

In Figure 17, the guidance, navigation and control side of the information flow represents the control engineers view of the product line instantiation problem while the resource scheduling and allocation side of the information represents the computer science view of the problem.

The technical themes of the Honeywell effort are:

- Architectures based on formal models,
  - accurate analytic/what-if capabilities
  - design synthesis/optimization capabilities
  - scalable/parametric architectures
  - abstract specification languages
  - enhanced reliability, reduced verification and validation
- Multiple integrated views
  - comprehensive requirements coverage
  - skill-specific views
  - requirements-specific views

- support for system trade-offs & optimization

• Layered, scalable architectures and open toolset to enhance reuse in related domains.

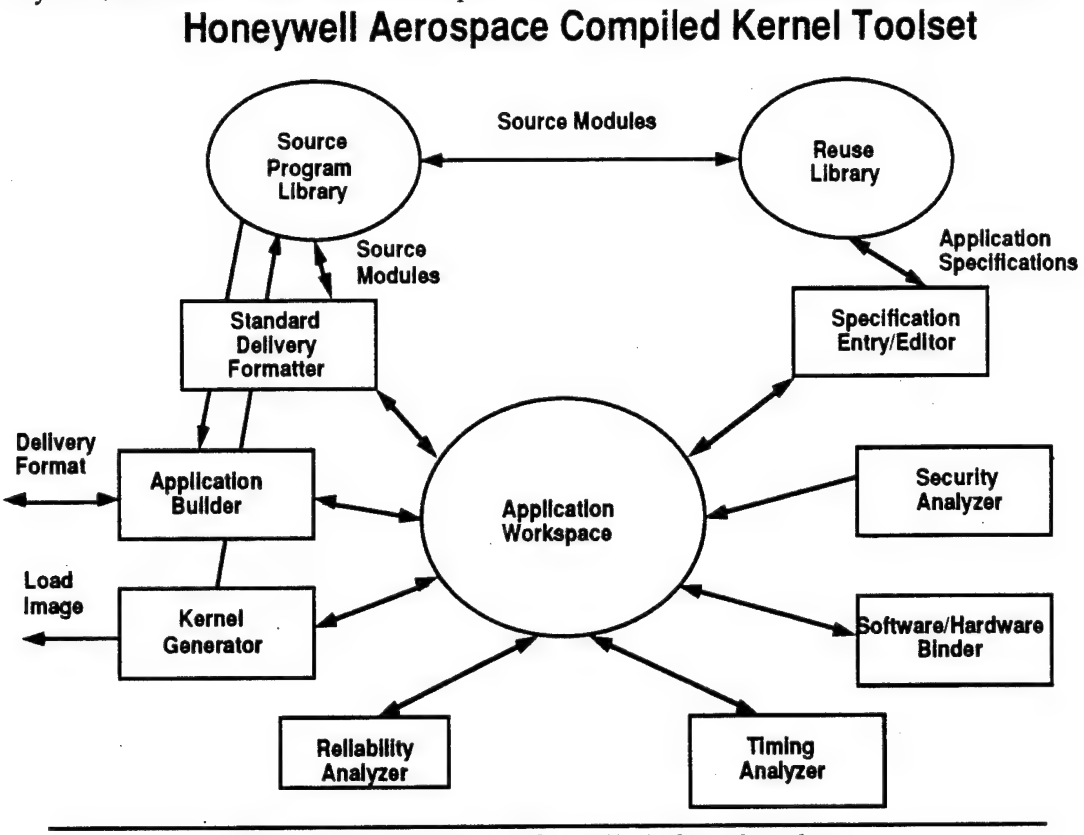


Figure 18. Honeywell Compiled Kernel toolset

The Honeywell approach is to initially use a ControlH toolset, where the primary Architectural Representation Language is block diagrams to capture the control architecture. The ControlH language features: mixed visual/textual (common user interface), connection (declarative) language, generic operators, structured/hierarchical operators, type inferencing, switches and conditionals, nonlinear look-up functions, first class operators and timing and implementation specifications. A Compiled Kernel toolset is being created to aid in the instantiation of the control architecture in a specific configuration (see Figure 18).

It is intended that the Compiled Kernel toolset will produce a multiprocessor load image meeting the application specification. Honeywell does not claim to have a proposed graphical syntax, just a convenient pseudo-graph notation. Specifically, ControlH provides a way to perform functional parameterization and specification of the execution path. The execution path can be represented graphically as a sequence of blocks to be executed. The sequence of blocks represent a process. Processes can be executed concurrently but modules are sequential and modes are exclusive. Processes can share data through monitors and pass data through messages. Processes can be periodic or aperiodic. Movement from a periodic to an aperiodic process is determined by an event (see Figure 19). Experiments are conducted to determine the role played by "mode switching."

Honeywell Modes and Process Sharing

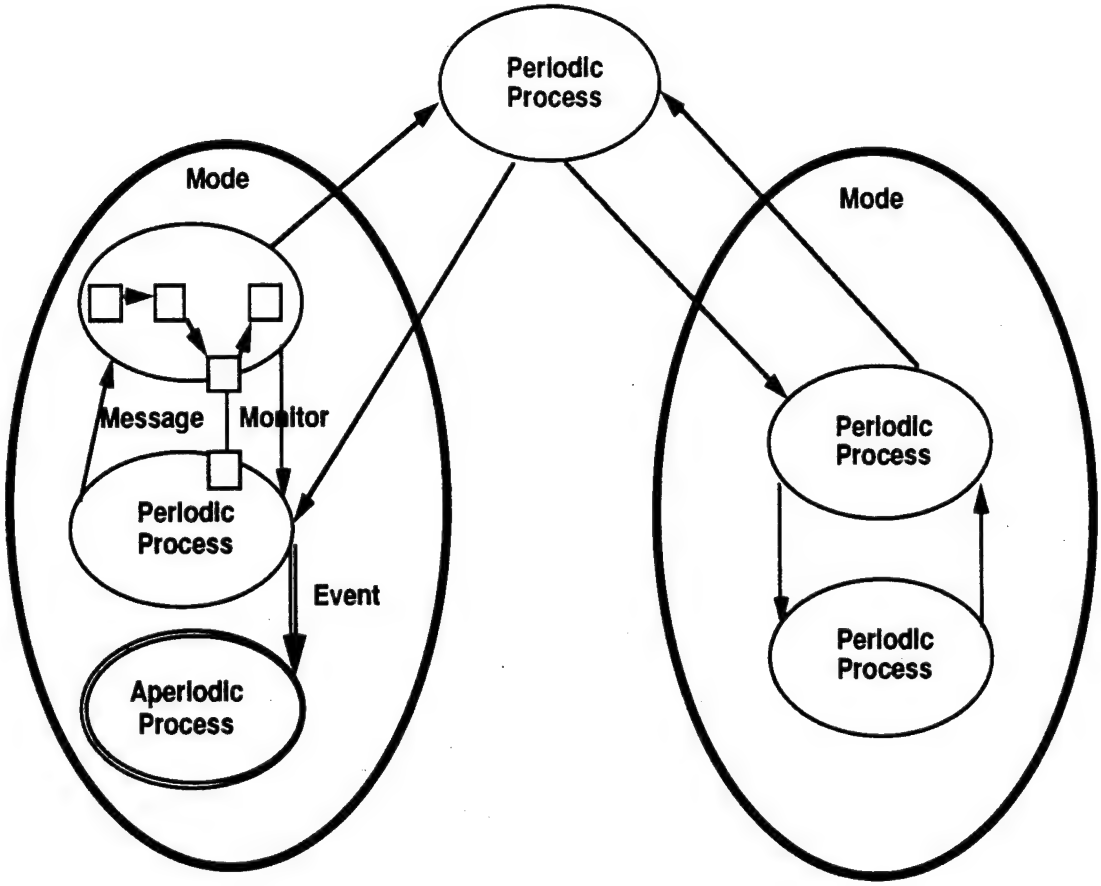


Figure 19. Honeywell modes and process sharing

Mode switching supported by the Honeywell Compiled Kernel Toolset is fully synchronous and is a mechanism (no data exchange) to enable fundamental changes in processing such as:

- Change in the number of processors, or
- Change in sampling rate.

Honeywell sees future reliable real-time issues as:

- Scheduling
  - static hard-real-time communication and event scheduling
  - separate output and state update preemptive scheduling
  - aperiodic communications
  - discrete-event dynamic systems
    - √ event sequencing and timing specifications
    - √ queuing and stochastic performance
- Reconfiguration
  - reactive systems/adaptive scheduling
  - on-line reconfiguration and load shifting
- Verification & Validation / Certification
  - component verification reuse
  - architecture verification

- automated verification
- Execution path and timing analysis & estimation

The Honeywell tools provide extensive support for modeling and implementing switching-mode control. Deciding when to switch between modes of control remains a matter of experimentation and engineering heuristics.

3.2.3 Discussion of Defense Information System Agency Architecture

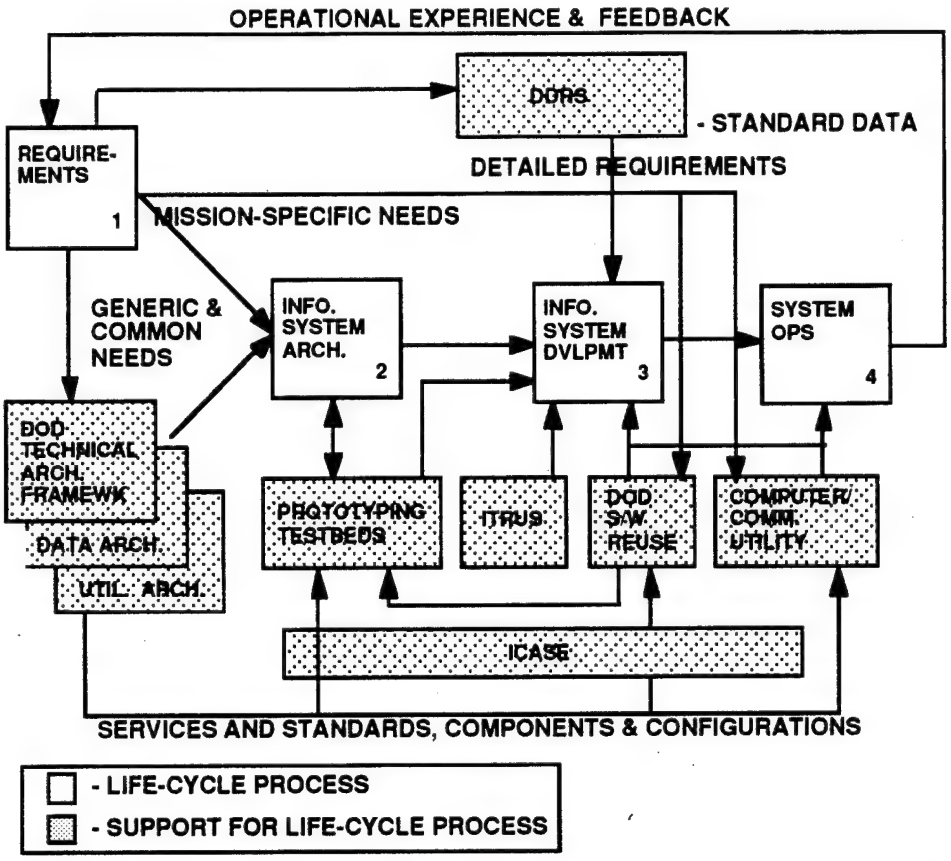


Figure 20. Technical Architecture Framework for Information Management (TAFIM)

The Department of Defense Technical Architecture Framework for Information Management (TAFIM) is the guidance for future information system development through use of Integrated Computer-Aided Software Engineering (ICASE) tools, application of Information Technology for Reuse (ITRUS), reuse of software stored in the Defense Software Repository (DSRS), and application of standard data elements stored in the Defense Data Repository (DDRS). The TAFIM requires repeated application of reference architectures to provide generic and common needs as part of the support for software life-cycle processes. The TAFIM is the top-level guidance for evolution of DoD enterprise (business) software and has been recommended for use in planning for the (national-scale) Advanced Distributed Simulation (ADS) system being constructed to support DoD in planning for a broad range of defense functions, including: (1) training and readiness, (2) requirements definition, (3) prototyping, (4) program planning, (5) design and manufacturing, and (6) test and evaluation. The Defense Information Systems Agency (DISA) is currently using the IDEF-0 and IDEF-1 modeling approaches for creating business process models. Unfortunately IDEF does not support capture of continuum processes. Thus the models must be augmented in some fashion to be used for ADS.

3.3 Development of supporting technologies for enterprise integration:

3.3.1 Enterprise Integration Technology (EIT):

EIT is a small business that was formed as a spinoff from Stanford University and is focused on the creation of network solutions for enterprise integration. A consortium led by EIT has recently won a Technology Reinvestment Program (TRP) award to develop infrastructure for use of the National Information Infrastructure (NII) to support enterprise integration.

3.3.2 Enterprise Integration Network (EINet):

Two years ago the Microelectronics and Computer Corporation (MCC) formed the Enterprise Integration Network (EINet) to facilitate creation of virtual companies. The intent is to provide network services to support small businesses to bid on requests for proposals which they would otherwise not be aware of or able to pursue. The enabling mechanism is to use networks to rapidly share information on opportunities, form transitory alliances for the purpose of answering the requests for proposals, maintain the alliance during execution of the resulting contract, and dissolve the alliance upon completion of the work. The MCC goal is to eventually support all 250,000 small businesses in the US.

3.3.3 Open Software Foundation Distributed Computing Environment (OSF DCE):

The Open Software Foundation Distributed Computing Environment (OSF DCE) is a vendor-neutral distributed computing infrastructure standard which supplies essential distributed computing services necessary to provide enterprise integration at the highest levels in the enterprise. These services are:

- A directory service which can locate entities anywhere throughout the enterprise,
- A security service which can verify the identity and authority levels of users throughout the enterprise, and
- A communications service which allows users and applications to communicate with other applications or to access data throughout the enterprise.

Companies are offering tools to assist businesses in using DCE to implement client-server solutions for integration of business functions. For instance, The Open Environment Corporation offers the Surround product as a means of implementing the Remote Procedure Calls for client-server applications. OEC specifically recommends a three-tiered client-server architecture (Figure 21).

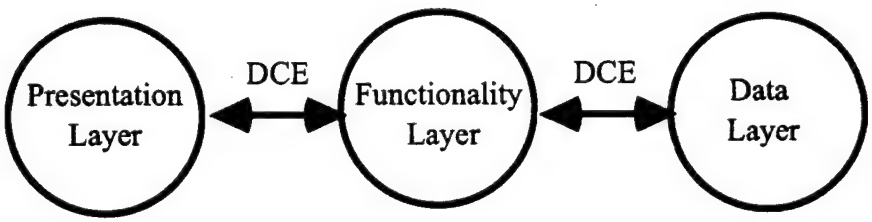


Figure 21. Three-tiered client/server architecture

3.4 Other hybrid systems research activities

Interest in hybrid systems has increased rapidly since John James and Robert Grossman coordinated invited sessions at IEEE Conferences on Decision and Control (CDC) in 1992 and 1993. Panos Antsaklis and Michael Lemmon subsequently organized an invited session on hybrid

systems for CDC '94. Anil Nerode organized a workshop on hybrid systems at Cornell in 1992 resulting in a Springer-Verlag text on hybrid systems. In 1993, MIT organized an ARO-sponsored workshop to discuss, among other topics, the differential geometry of Hybrid systems. At the MIT meeting, Wolf Kohn, Anil Nerode and Roger Brockett presented models for infinitesimal controls in the hybrid domain. In 1994, at Harvard , a follow-on workshop was held in which the computability issues of infinitesimal controls for hybrid systems via dynamic programming was discussed. Wolf Kohn, Anil Nerode, Michael Branicky and Albert Benveniste presented analysis of the chattering solution proposed by Kohn and Nerode in 1992. Wolf Kohn prepared lecture notes for the Graduate Course in Hybrid Control offered, for the first time, at UC, Berkeley by Pravin Varaiya and Shankar Sastry. Anil Nerode has organized this workshop at Cornell and another Springer-Verlag volume is expected. Program Committee members for this second meeting are: Anil Nerode, Panos Antsaklis, Amir Pnueli, Shankar Sastry, Pravin Varaiya, Wolf Kohn, and John James.

4. Overview of use of MAHCA for enterprise-wide control.

We have provided a detailed statement of the manufacturing planning, scheduling, and control problem in MAHCA terms. We have developed similar statements of complex computer-controlled systems for:

- Transportation,
- Cruise missile control,
- Crew station control ,
- Medical informatics,
- Communications,
- Military Command and Control,

In the next section we will provide a brief overview of the architecture development work being conducted for the military command and control problem.

5. Military Command and control.

The Army intends to field a digital division by 1998. Capabilities of the digital division are yet to be fully determined. However, it is expected that future Army combat operations will increasingly involve coalition forces and that future Army missions will increasingly require conduct of operations other than war (OOTW), such as peacemaking, peacekeeping, humanitarian support, and humanitarian relief. Current plans to reduce force structure is driving the Army to investigate increasing flexibility of existing units to support a wide range of missions.

Innovative use of barriers to decrease mobility of opposing forces and active use of barriers to cause opposing forces to move in desired directions has been a historic discriminator between success and failure in war. Barriers, such as intelligent mines are often used to channel opposing forces into an area where they can be engaged by direct and indirect fire weapons of the combined arms team. Successful demonstration of command and control of advanced mines will provide commanders with a flexible, lightweight means of increasing combat effectiveness. The Army experts in mine warfare determine the critical operational issues and criteria for success of the intelligent minefield. Future efforts to coordinate results of our current demonstration of target engagement with the intelligent minefield would provide an opportunity to test concepts for dominating the maneuver battle with concepts for operational use of intelligent mines.

Architecture Development: We have participated in a limited scope effort to demonstrate MAHCA in support of engaging multiple targets with multiple weapons platforms. The limited scope multiple agent demonstration has led to simplifications of the environment and the operational context. The intent is to demonstrate the major objectives of the Domain-Specific

Software Architecture (DSSA) program without being required to commit excessive resources to a high-fidelity model. Our specific objectives in the project have been to:

1. Expand the generic architecture previously developed for multiple-agent hybrid control of distributed, real-time processes
2. Create a reference architecture for engagement processes
3. Demonstrate on-line rescheduling of real-time processes
4. Demonstrate parameter adaptation of architecture parameters in the presence of model parameter uncertainty
5. Demonstrate structural adaptation (i.e. automatic adaptation of the sharing of system control between structurally different models) in the presence of model structural uncertainty
6. Demonstrate use of the Equational Reactive Inferencer syntax as an appropriate architectural description language for simultaneously capturing the logic and the dynamics of the target engagement domain.

Battlefield Environment Model:

The elements of the proposed demonstration are: (1) a Battlefield Environment, and (2) the controlling agents. We discuss these elements next.

The *battlefield environment* (see Figure 22) consists of a Universe (a prespecified region of the plane, i.e. a closed surface of $R^{2)}$, two types of objects (Friendly objects and Foe objects), and the rules which determine the reaction (the evolution of the state over time) of friend or foe objects given the current state (of friend and foe objects) and the context of the operational situation (set of logical inputs). The set of possible battlefield scenarios is the set of **sequences of friend and foe actions** (from initiation of the battlefield simulation until the friend and foe objects are in a terminal state) and **associated contexts**. For the purposes of this demonstration, we will have a limited set of friendly tank objects (always three objects) and foe objects (initially three tanks and two scouts) and a limited number of rules (equational clauses - see Section 5.1.1 and [44]) which determine the evolution of the state of the system. The model is deliberately constructed in order to reflect some of the actual conditions which occur on the battlefield but only at a level necessary to demonstrate the reasoning and behavioral capabilities of multiple-agent hybrid control. Specifically, we insert a fourth control agent to provide supervisory control of the three tanks. Supervisory control actions are limited to assignment of sectors of fire and allocation of targets to engage (actual decisions to fire are made by the local control agents for each tank). For purposes of the demonstration we simulate results of engagement decisions but in an actual system the program would be used in an advisory role.

While the structure of the demonstration and the scope of multiple-agent hybrid control theory admit the construction of a high-fidelity model we will *not achieve* that in this demonstration. What we *will achieve* is a unification of logic and evolution equations describing the engagement of multiple targets by multiple weapons. The logic equations often used by computer scientists to define finite state machines ignore system dynamics and the differential operator equations used by engineers ignore system logic. We apply an established optimization procedure (relaxed variational control - see [13]) to the system of equations describing our hybrid control architecture to extract an optimization automaton after appropriate embedding of the logic equations in the continuum representation. This result will demonstrate the on-line extraction of automata which implement distributed, intelligent, real-time control. The simple model is discussed below.

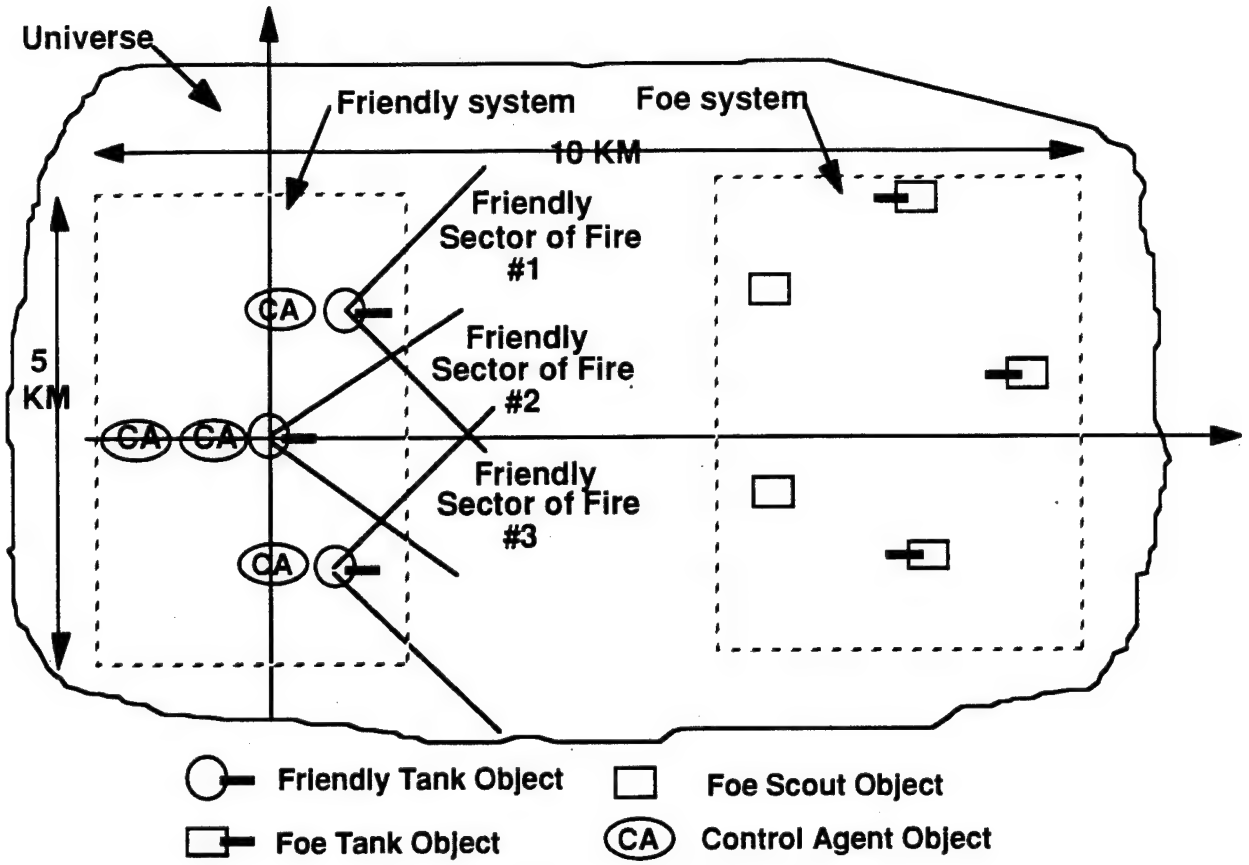


Figure 22. Battlefield Environment

Friendly command and control is limited to one commander (a Control Agent) making decisions based on battlefield sensors (Information Gatherer) and (noisy) information concerning friend and foe situation. Friendly weapons are fixed and consist of direct fire weapons or smart mines (both labeled Attack System). Information concerning enemy activity is obtained from battlefield sensors aggregated into an agent labeled Information Gatherer. Thus, there are three friendly, hierarchically organized objects; two low-level objects each labeled Attack System and a high level object labeled Information Gatherer.

In the scenario there are two types of enemy objects. Both types of enemy objects are crew-served vehicles. These vehicles are labeled Scout objects and direct-fire weapon systems (labeled Active objects) respectively. There is a variable number of foe objects (this number is specified at the beginning of a demonstration).

We will refer to the collection of three friendly objects as the Friendly System and the collection of foe objects as the Foe System.

The goal of the Friendly System is to coordinate the use of its assets (the three Attack systems. See Figure 22) to maximize the damage (Kill Probability) inflicted on the Foe System with admissible survivability (Survival Probability). The goal of the foe system is to detect the assets of the friendly system and to destroy them The location of each of the Friendly objects in the coordinates of the universe is fixed. Each of the Foe objects follows *prespecified* (autonomous) trajectories in the Universe. friendly system acquires knowledge about position and perhaps the velocity of each of the Foe Objects via noisy sensors.

The Friendly System is controlled by a MAHCA controller composed of three agents. Each of the objects in the Friendly System, the two Attack Systems and the Information Gatherer are controlled by a MACHA agent. The battlefield scenario is illustrated in Figure 22.

We conclude this overview of our proposed model with a discussion of the *script* we are developing for the demonstration. The general idea is for the agents to implement on-line a coordinated Reactive, adaptive policy [44] to 'win' the battle. This means achieving the highest kill probability over the prespecified duration of the script.

The actions of the control agent of each of the Attack objects are of two types : *tracking*, via a noisy sensor, of a particular foe object, *selected* by the control agent of the Information Gatherer, and command *firing* against the foe object under tracking. The Information gatherer decision domain includes an action to *transfer* the tracking action of a foe object to tracking another one. In an attack agent, the decision to *engage* a foe object is solely a function of the relation between the local kill and survival probabilities relative to the foe object under tracking and the global kill and survivabilities of the friendly system inferred by the information gatherer control agent. The demonstration will have two level of agents with three agents controlling the three platforms in the simulation and a fourth MAHCA agent coordinating the activities of the other three. While consideration has been given to having one or more control agents coordinate primarily with a smart mine agent, the simulation will model three tanks as the friendly platform agents. Future expansions of the demonstration will support one of the MAHCA tank controller agents sending high-level guidance to the single-agent controller of a test fixture at Picatinny Arsenal. The major elements of the demonstration are: (1) A simulation of the battlefield environment comprised of friend and foe systems (running on a SPARC-10?), (2) An interface between the battlefield simulation and the MAHCA agents, (3) MAHCA agents running on separate 586 machines, (4) the single-agent controller performing low-level control under the guidance of a MAHCA agent, (5) LabView as the interface software to enable the sensors and actuators. The engagement scenario is summarized below.

Battlefield Environment (Engagement Scenario):

1) Five targets at 5 locations going on 5 different paths:
   a). Tank on a crossing pattern
   b) Scout on a diagonal, jinking pattern
   c) Tank on a jinking, inbound pattern
   d) Helicopter hovering and slow crossing
   e) Helicopter on a jinking pattern inbound and diagonal

2) Three platforms at three fixed locations with three different sensor suites and capability suites:
   a) Tank with suite a
   b) Tank with suite b
   c) Tank with suite c

3) The three platforms are controlled by three agents

   The three agents are coordinated by a higher-level agent

One of the three agents obtains input from one of the platforms in the simulation world and sends appropriate commands to a single-agent controller which controls the test fixture.

4) An engagement scenario consists of:

a) Targets moving on preassigned patterns. Some targets may be capable of detecting observation by active sensors. All targets will certainly detect engagement by friendly forces and may elect to engage or disengage.

b) Each of the three platforms:
(1) Detect targets using onboard sensors
(2) Identify targets
(3) Provide data to a controlling agent
(4) Select target to engage or are assigned a target by the coordinating agent
(5) Track selected target
(6) Engage target
(7) Assess engagement and reengage or select another target

5) Sensors need to be simulated in the battlefield environment. While data was provided for optical sensors, provision should be made in the data structure for other types of sensors:
a) Optical including
(1) Optical magnification
(2) Detection range
(3) Recognition range
(4) Identification range
(5) Field of View
b) Thermal (data similar to optical)
c) Millimeter wave (data similar to optical)
d) Once target is detected, assume sensor provides azimuth and range information

6) Nominal pointing parameters:
a) Maximum acceleration
b) Maximum slew speed
c) Minimum smooth tracking slew speed (to avoid stiction)
d) Pointing accuracy (see Figure 23)
e) Use nominal parameters to allocate error budget
f) Data provided for ATAC, M1A1, Leopard, Centurion, and Apache

7) Total system error data:
a) Engagement accuracy can be divided into two elements: dispersion and bias
(1) Dispersion data accounts for differences in wind, ammunition, statistical deviations in system performance
(2) Aiming bias can throw off engagement accuracy much more unexpectedly than aiming dispersion. Aiming bias increases with maneuvering target and distance to target (increased time of projectile flight)

8) Angular acceleration data influences aiming bias. Can get angular acceleration data from tracker (the test fixture or the simulated sensor) but it is HARD to get accurate numbers for a maneuvering target (double the standard deviation for a non-maneuvering target).

9) Assume a circular target for calculation of probability of hit. Translate angular error data into a hit probability.

10) The interface is very clean between the multiple agents and the simulation of the platforms and the targets in order to be able to use the interface solution to drive other simulation software.

11) Once a platform has fired, the subsequent hit probabilities go way down since the target will begin maneuvering.

12) The simulation begins with targets coming into the detection range of the sensors and ends with the attackers or defenders being eliminated.

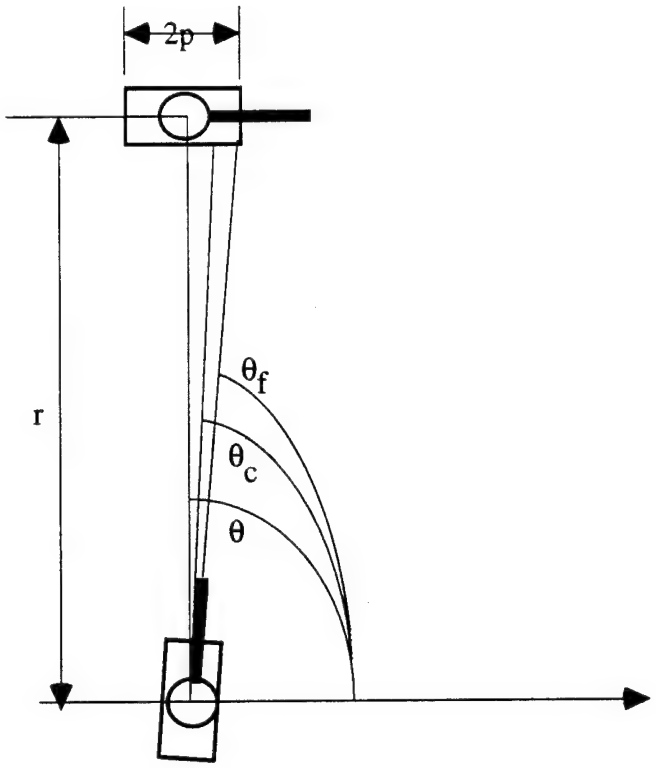


Figure 20. A foe object engaging a friendly object

6. Summary

We have developed a canonical representation of interacting networks of controllers. Given a connectivity graph with N nodes (controllers) and the corresponding agent's knowledge bases, a network of 2N agents can be constructed with the same input-output characteristics, so that each agent interacts only with another (equivalent) companion agent, whose knowledge base is an abstraction of the knowledge in the network. Thus, in general, the multiple-agent controller for any network configuration is reduced to a set of agent pairs.

One agent of the agent pair maintains coordination with other agent pairs across the network. We call that agent of the pair which represents network information the Thevenin Agent, after the author of a similar theorem in electrical network theory. The proof carried out by the Thevenin Agent generates, as a side effect, coordination rules that define what and how often to communicate with other agents. These rules also define what the controller needs from the network to maintain intelligent control of its physical plant.

Our approach develops a canonical way to prove the theorem characterizing the desired behavior for each agent by constructing and executing on-line a finite state machine called the "proof automaton." The inference process is represented as a recursive variational problem in which the criterion is an integral of a function called the Generalized Lagrangian. The Generalized Lagrangian maps the Cartesian product of equational rules and inference principles to the real line, thus effectively providing a hill-climbing heuristic for the inference strategy of the theorem prover. In MARC the inference steps play a role analogous to action signals in conventional control, while vector fields on the manifold M constitute generators of feedback laws.

References

[1] J. S. Albus, C. McLean, A, Barbera, M. Fitzgerald, "An Architecture for Real-Time Sensory-Interactive Control of Robots in a Manufacturing Environment," 4th IFAC/IFIP Symposium on Information Control of Robots in Manufacturing Technology, Gaithersberg, MD, Oct. 1982.

[2] A.J. Barbera, J.S. Albus, M.L. Fitzgerald, and L.S. Haynes, "RCS: The NBS Real-Time Control System," Robots 8 Conference and Exhibition, Detroit, MI, June 1984.

[3] J.S. Albus, H.G. McCain, and R. Lumia, "NASA/NBS Standard Reference Model for Telerobot Control System Architecture (NASREM)," NIST (formerly NBS) Technical Note 1235, April 1989 Edition.

[4] J.S. Albus, andR. Quintero "Toward a Reference Model Architecture fir Real-Time Intelligent Control Systems (ARTICS)," Proceedings of the Workshop on Software Tools for Distributed Intelligent Control Systems, Charles J. Herget ed. September 1990.

[5] J.S. Albus, R. Quintero, R. Lumia, M. Herman, R. Kilmer, and K. Goodwin, "Concepts for a Reference Model Architecture for Real-Time Intelligent Control Systems (ARTICS)," NIST Technical Note 1277, April, 1990.

[6] R. Quintero, "Toward an Intelligent Control Systems Development Methodology," Proceedings of the JSGCC Software Initiative Workshop, 1-4 December 1992. Guidance and Control Information Analysis Center (GACIAC) PR92-03.

[7] Kohn, W., J. James, and A. Nerode, "Multiple-Agent Reactive Control of Distributed Interactive Simulations (DIS) Through a Heterogeneous Network," Proceedings of the Army Research Office Workshop on Hybrid Systems and Distributed Interactive Simulations, Research Triangle Park, NC 28 Feb. - 1 Mar. 1994.

[8] Kohn, W., J. James, A. Nerode, K. Harbison, and A. Agrawala, "A Hybrid Systems Approach to Computer-Aided Control System Design," Proceedings of the Joint Symposium on Computer-Aided Control System Design, Tucson, AZ, 7-9 March 1994.

[9] Nerode, A., and W. Kohn, "Multiple Agent Declarative Control Architecture", *Hybrid Systems*, Springer Verlag, 1993, Volume 736, R. L. Grossman, A. Nerode, T. Rischel, A. Ravn, eds.

[10] Kohn, W and A. Nerode, "Models for Hybrid Systems: Automaton, Topologies, Control, Controllability, Observability", *Hybrid Systems*, Springer Verlag, 1993, Volume 736, R. L. Grossman, A. Nerode, T. Rischel, A. Ravn, eds.

[11] Computer Science and Telecommunications Board, National Research Council, *Realizing the Information Future: The Internet and Beyond*, National Academy Press, Washington, D.C., 1994.

[12] Nerode A., Kohn W., " Multiple Agent Autonomous Control: A Hybrid Systems Architecture" Logical Methods In Honor of Anil Nerode's Sixtieth Birthday, N. C. Crossley , J. B. Remmel, M. E. Sweedler, Eds., Birkhauser, Boston, 1993.

[13] Kohn W. and Nerode A. "Multiple Agent Hybrid Control Architecture" MSI Report 93-11 Cornell U

[15] Young, L.C. "Optimal Control Theory" Chelsea Publishing Co., NY, 1980.

[16] Nerode A. and W. Kohn " An Autonomous Systems Control Theory: An Overview" Proc. IEEE CACSD'92, March 17-19 Napa Ca.

[17] Kohn, W. "Declarative Hierarchical Controllers" Proceedings of the Workshop on Software Tools for Distributed Intelligent Control Systems, pp 141–163, Pacifica, CA, July 17–19, 1990.

[18] Kohn, W. and T. Skillman "Hierarchical Control Systems for Autonomous Space Robots" Proceedings of AIAA Conference in Guidance, Navigation and Control, Vol. 1, pp 382–390, Minneapolis, MN, Aug. 15–18, 1988.

[19] Kohn, W. "A Declarative Theory for Rational Controllers" Proceedings of the 27th IEEE CDC, Vol. 1, pp 131–136, Dec. 7–9, 1988, Austin, TX.

[20] "DSSA Guidelines Draft 0.1", Proceedings of the DSSA Workshop IX, Teknowledge Federal Systems, MD, 28-29 March, 1994.

[21] Kohn, W., J. James, A. Nerode, and J. Lu "Multiple-Agent Hybrid Control Architecture for the Target Engagement Process (MAHCA-TEP - Version 0.2 of MAHCA-TEP: Technical Background, Simulation Requirements, and Engagement Model", Intermetrics Technical Report, 25 August 1994.

[22] Kohn, W. "Rational Algebras; a Constructive Approach" IR&D BE-499, Technical Document D-905-10107-2, July 7, 1989.

[23] Kohn, W. "Declarative Control Architecture" CACM Aug 1991,Vol34, No8.

[24] Skillman, T., W. Kohn, et.al. "Class of Hierarchical Controllers and their Blackboard Implementations" Journal of Guidance Control & Dynamics, Vol. 13, N1, pp 176–182, Jan.–Feb., 1990.

[25] Lloyd, J.W. "Foundations of Logic Programming" second extended edition, Springer Verlag, NY, 1987.

[26] Kohn, W. "The Rational Tree Machine: Technical Description & Mathematical Foundations" IR&D BE-499, Technical Document D-905-10107-1, July 7, 1989.

[27] Mesarovic, M. and Y. Tashahara "Theory of Hierarchical Multilevel Systems" Academic Press, NY, 1970.

[28] Kowalski, R. "Logic for Problem Solving" North Holland, NY, 1979.

[29] Robinson, J.A. "Logic: Form and Function" North Holland, NY, 1979.

[30] Kohn, W. "Rational Algebras; a Constructive Approach" IR&D BE-499, Technical Document D-905-10107-2, July 7, 1989.

[31] Kohn, W. "The Rational Tree Machine: Technical Description & Mathematical Foundations" IR&D BE-499, Technical Document D-905-10107-1, July 7, 1989.

[32] Skillman, T., W. Kohn, et.al. "Class of Hierarchical Controllers and their Blackboard Implementations" Journal of Guidance Control & Dynamics, Vol. 13, N1, pp 176–182, Jan.–Feb., 1990.

[33] Kohn, W. "Application of Declarative Hierarchical Methodology for the Flight Telerobotic Servicer" Boeing Document G-6630-061, Final Report of NASA-Ames research service request 2072, Job Order T1988, Jan. 15, 1988.

[34] Kohn W. " Multiple Agent Inference in Equational Domains Via Infinitesimal Operators" Proc. Application Specific Symbolic Techniques in High Performance Computing Environment". The Fields Institute, Oct 17-20 1993.

[35] Kohn W., and Nerode A., "Multiple- Agent Hybrid Systems" Proc. IEEE CDC 1992, vol 4, pp 2956, 2972.

[36] Hoard, J., and W. Kohn, "A Relational Model of Natural Language Semantics", to be released as a Boeing Technical Report.

[37] Kohn, W. "Distributed Hybrid Controller Architecture," Proceedings of Hybrid Systems and Autonomous Control, Mathematical Sciences Institute, Cornell University, Ithaca, NY, 28-30 Oct, 1994.

[38] Landauer, C, and K. L. Bellman, "New Mathematics for Computing (a new initiative)," Proceedings of Hybrid Systems and Autonomous Control, Mathematical Sciences Institute, Cornell University, Ithaca, NY, 28-30 Oct, 1994.

[39] Mettala, E. G., James, J. R., Coleman, N., Gallagher, E. J., Harris, R. L., Smith, J. G., and Graham, M. "Domain-Specific Software Architectures: Government Needs and Expectations." Presented at the IEEE Symposium on Computer-Aided Control System Design, Napa, CA 17-19 March, 1992.

[40] Tracz, W., and Coglianese, L. "An Adaptable Software Architecture for Integrated Avionics" ADAGE-IBM-93-03. IBM Federal Systems Division.

[41] Hayes-Rith, F., Erman, L. D., Terry, A., and Hayes-Roth, B., "Distributed Intelligent Control and Management (DICAM) Applications and Support for Semi-Automated Development." Proceedings of AAAI-92 Workshop on Automating Software Development, San Jose, CA, 1992.

[42] Vestal, S. " Integrating Control and Software Views in a CACE/CASE Toolset," Proceedings of the Joint IEEE/IFAC Symposium on Computer-Aided Control System Design, Tucson, AZ, 6-9 March, 1994.

[43] Report, Defense Science Board Task Force on Simulation, Readiness and Prototyping, Dr. Joseph V. Braddock and General Maxwell R. Thurman, Co-Chairmen, 21 December 1992.

[44] Teknowledge Federal Systems, "DSSA Guidelines Draft 0.1," Proceedings of the DSSA Workshop IX, 28-29 March, MD, 1994.

[45] RIG Proposed Standard "A Uniform Data Model for Reuse Libraries (UDM)," Reuse Library Interoperability Group, 20 January, 1994.

[46] RIG Technical Report "Measuring Reuse Library Interoperability: Applying the GQM Paradigm," Reuse Library Interoperability Group, 20 January, 1994.

[47] Kohn, W., J. James, and A. Nerode, "Multiple-Agent, Hybrid Control Architecture: A Generic, Open Architecture for Incremental Construction of Reactive Planning, Scheduling and Control Systems for Manufacturing," Intermetrics White Paper, October 1994.

[48] Kohn, W., A. Nerode and J. Remmel, "Multiple-Agent Control of Hybrid Systems," March, 1994.

Appendix A: Equational Logic Language

Hybrid Systems Language Requirements: A formal language that encodes hybrid systems must be able to express: evolution models, logic models, interfaces of evolution and logic models, behavior requirements, and real time constraints.

Evolution Models: An evolution model is a composite of 3 items: An Evolution Domain, a Set of Transformations on that domain and Boundary Conditions.

- *Evolution Domain:* The domain of a Hybrid system is always a Manifold, called the Carrier Manifold or domains derived from it such as vector bundles, jet spaces, congruence manifolds etc. (vector spaces are the simplest examples of manifolds). A manifold is a set of points S together with a countable set of maps, called Coordinate Maps. The domain of each coordinate map is an open subset of S, the range is an open subset in a Euclidean space. The coordinate maps are specified by a set of Generic and Particular properties. The generic properties are common to all manifolds. The particular properties describe characteristics of each application. Given these properties and the set S the manifold is completely determined. Thus, a language expressing hybrid systems must provide primitives for encoding sets and coordinate transformation properties. Moreover it must provide means to express domain items that depend on the manifold (ex: tangent spaces, vector bundles, submanifolds, immersions, jet spaces, products of manifolds, etc.).
- *Transformation Set:* A transformation is an item that transforms a point or points in the manifold to another point or points. Examples include differential (ordinary or partial) equations, difference equations, a stochastic process, integral equations, algebraic transformations, a variational expression, an algorithm, a set of rules and conjunctions or disjunctions of the above. Without loss of generality, a set of transformations generate Trajectories in the carrier manifold or in its associated domains. Thus, the language of hybrid systems must be able to express transformations, and the outcome of transformations (namely trajectories). Note that in general trajectories are infinite objects so the language must exhibit the capabilities to express infinite objects with finitary representations (example; flows).
- *Boundary Conditions:* These are standard initial or constraint items for the transformations defined above. They define subdomains within the manifold or its associated domain items.

Logic Models: A logic model is an object composed of three items: A Logic Domain, an Axiom Base and A Proof or Discrimination System.

- *Logic Domain:* The domain of the logic model are certain classes of trajectories of evolution models on the carrying manifold called *relaxed curves* The state of a hybrid system and its behavior are relaxed curves. These curves must possess certain properties (If the hybrid system is a model of a control problem these properties are requirements). A language for hybrid systems must have primitives for encoding, manipulating, connecting and transforming Relaxed Curves. In the Kohn- Nerode theory a central result states that relaxed curves can be expressed by one or more strings of the form q(X, ...,Y) *rel* p(X, ...,Y) with p and q algebraic expressions in a certain algebra and *rel* one of three relational connectives: equal =, not equal $\neq$, and partial order $\leq$.
- *Axiom Base*: The Axiom Base is an encoding of the properties listed above, together with the axioms of algebraic logic with equality, and the axioms of properties of the logic operators (and, or , <=, not) and the properties of certain class of operators, called Inference operators that transform elements of the domain into other elements of the domain. For example unify, which implements Robinson unification, transforms a set of relaxed curves into a relaxed curve whose properties are common to the set. The Axiom base also encodes properties of the hybrid system as a whole (global properties) The central ones are Stability, Controllability, Goal Reachability, Observability, and Self-Awareness.

- *Proof System:* The proof System is a formal representation of the "Intelligence" of the hybrid model. It uses the axiom base to generate Relaxed curves that correspond to the behavior of the system. It encodes a *strategy* (Monotone Dynamic Programming in the Kohn Nerode Theory) for the effective construction of relaxed curves that characterize the behavior of the hybrid system if they exist or Modification instructions (to the model) if they do not exist.

Interface: The interface between Evolution Models and Logic models characterizes the central property for well-posedness of hybrid models: *continuity* of behavior trajectories (relaxed curves) in the carrier manifold. By continuity we mean that the Transformation that maps the requirements encoded in the axiom base to the space of behavior trajectories in the manifold or its derivatives is continuous. When one talks about continuity, one must specify a topology; In hybrid systems the topology is the topology of the carrier manifold which is fully determined once the coordinate maps are defined. In our approach to hybrid systems the axiom base must define the coordinate maps at each instant. A formal language for hybrid systems must contain primitives and a composition strategy for expressing desired continuity definition. For example, in the commercial airplane problem, desired continuity is that the behavior trajectories be such that "a coffee cup 3/4 full in any location of the airplane should not spill (relative topology)."

Behavior Requirements: Language requirements for a scenario-based approach for determining system requirements and expressing these results in a formal language have been studied. Descriptions about primitives for expressing scenario-based behavior requirements are summarized in [17]. Also, these primitives are described in the Hybrid Systems book recently published. However we emphasize here two primitives that the behavior requirements must specify; these are the Generalized Lyapunov stability check and the goal reachability check. Both of these can be written as equations with variables in the space of relaxed trajectories. Each of these equations have non-empty solution sets only if the corresponding properties are satisfied.

Real Time constraints: These are equational forms that characterize the computer environment. A language for hybrid systems must have primitives that express real-time frames, infinitesimal concurrency, relaxation level, "sufficiently good paradigms", any time computing, etc. These real-time constraint primitives must be expressible in terms of *continuity*.